z/OS Communications Server

IP Configuration Guide

Version 1 Release 4

SC31-8775-02

z/OS Communications Server

IP Configuration Guide

Version 1 Release 4

SC31-8775-02

Note:
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 813.

Third Edition (September 2002)

This edition applies to Version 1 Release 4 of z/OS (5694-A01) and Version 1 Release 4 of z/OS.e (5655-G52) and to all subsequent releases and modifications until otherwise indicated in new editions.

Publications are not stocked at the address given below. If you want more IBM® publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Software Reengineering
Department G7IA/ Bldg 503
Research Triangle Park, NC 27709-9990
U.S.A.

If you prefer to send comments electronically, use one of the following methods:

Fax (USA and Canada):
1-800-254-0206

Internet e-mail:
usib2hpd@vnet.ibm.com

World Wide Web:
http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

IBMLink™:
CIBMORCF at RALVM17

IBM Mail Exchange:
tkinlaw@us.ibm.com

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures xv

Tables xix

About this document xxi
Who should use this document xxi
Where to find more information xxi
Where to find related information on the Internet xxi
Accessing z/OS licensed documents on the Internet xxii
Using LookAt to look up message explanations xxiii
How to contact IBM service xxiii
z/OS Communications Server information xxiii

Summary of changes xxxi

Part 1. Base TCP/IP system 1

Chapter 1. Configuration overview 3
z/OS TCP/IP stack function support 3
z/OS msys for Setup and Wizard 6
Wizard 6
z/OS msys for Setup 7
z/OS UNIX System Services (z/OS UNIX) concepts 10
Overview of data sets and HFS files 11
Hierarchical File System concepts 11
Understanding resolvers 12
Setting up a resolver address space 14
Resolver customization 15
Managing the resolver address space 18
Understanding search orders of configuration information 18
Configuration data set naming conventions 19
Configuration files for the TCP/IP stack 25
PROFILE.TCPIP search order 25
TCPIP.DATA search order 26
Configuration files for TCP/IP applications 26
Resolver configuration files 27
MVS-related considerations 37
MVS system symbols 37
Automatic restart manager (ARM) 38
Logging of system messages 39
Accounting - SMF records 40
Security considerations 44
UNIX System Services security considerations 45
Requirement for an OMVS segment 45
Authorization of TCP/IP started task user ID 47
Other user IDs requiring z/OS UNIX superuser authority 48
BPX.DAEMON facility class 48
Program control 49
Defining TCP/IP as a UNIX System Services physical file system (PFS) 50
References 52
Performance considerations 52
Fast path support 52
Considerations for multiple instances of TCP/IP 54

Common INET physical file system (CINET PFS) 54
Port management overview 55
Selecting a stack when running multiple instances of TCP/IP 60
Specifying BPXPRMxx values for a CINET configuration 64
Considerations for Enterprise Extender 65
Considerations for VIPA 65
Required steps before starting TCP/IP 67
Planning your installation and migration 67
Step 1: Install z/OS CS 68
Verifying the initial installation 68
Step 2: Customize z/OS CS 69
Step 3: Configure VMCF and TNF 72
Step 4: Update the VTAM application definitions 75
Step 5: Verify that the resolver address space is active 76
Step 6: Start the TCP/IP address space 76
Step 7: Set up cataloged procedures and configuration data sets 76
Step 8: Customize TCP/IP messages 76

Chapter 2. Security 79
System resource protection 79
Application security 79
TCP/IP resource protection 81
Protecting data in the network 87
Network security principals 87
Network security protocols 89
Security Event Reporting 99
Integrated Intrusion Detection Services (IDS) 99

Chapter 3. Customization 101
Configuring the syslog daemon (syslogd) 101
Configuration statements 101
Starting and stopping syslogd 105
Offloading log files 106
Using syslogd for z/OS UNIX application programs 107
Usage notes 108
Diagnosing syslogd configuration problems 108
Configuring TCPIP.DATA 109
Use of TCPIP.DATA and /etc/resolv.conf 109
Creating TCPIP.DATA 109
TCPIP.DATA statements 110
Configuring PROFILE.TCPIP 110
Changing configuration information 111
Setting up TCP/IP operating characteristics in PROFILE.TCPIP 112
Setting up physical characteristics in PROFILE.TCPIP 115
Setting up reserved port number definitions in PROFILE.TCPIP 139
Setting up SAF Server Access Authorization (SERVAUTH) (optional) . . . 143
Configuring the local host table (optional) 143
Why configure a local host table? 143
Creating HOSTS.LOCAL site host table 144
Creating /etc/hosts 146
Creating ETC.IPNODES and /etc/ipnodes 146
Verifying your configuration 148
Verify TCPIP.DATA and TCPIPJOBNAME 148
Verify /etc/resolv.conf 148
Verifying PROFILE.TCPIP with netstat or onetstat 149
Verifying interfaces with PING and TRACERTE 151

Verifying local name resolution with TESTSITE 152
Verifying PROFILE.TCPIP and TCPIP.DATA using HOMETEST 152
Verifying your X Windows System installation (Optional) 153

Chapter 4. Routing 155
Routing terminology 155
General terms 155
Interior Gateway Protocols (IGP) 156
Static versus dynamic routing 157
The sample network 157
IPv4 static routing 158
Using static routing with OMPROUTE 160
IPv6 static routing 161
Using static routing with router advertisements 162
Static routing configuration examples 162
z/OS TCPCS4 162
z/OS TCPCS7 163
IPv4 dynamic routing 165
Routing daemons 165
Migration from OROUTED to OMPROUTE 165
Dynamic routing using OMPROUTE 166
Configuring OSPF and RIP 179
IPv6 dynamic routing 205
Verification of routing (Static and dynamic) 205
Verifying connections with NETSTAT, PING, and TRACERTE 206

Chapter 5. Virtual IP Addressing 209
Terminology 209
Introduction to VIPA 209
Moving a VIPA (For TCP/IP outage) 211
Static VIPAs, Dynamic VIPAs (DVIPAs), Distributed DVIPAs 212
Using static VIPAs 213
Configuring static VIPAs for a z/OS TCP/IP stack 213
Configuring static VIPAs for Enterprise Extender 214
Considerations when using static VIPAs with IPv6 215
Planning for static VIPA Takeover and Takeback 215
Using Dynamic VIPAs (DVIPAs) 215
Configuring Dynamic VIPA (DVIPA) support 215
Planning for Dynamic VIPA Takeover 216
Different application uses of IP addresses and DVIPAs 218
Configuring Dynamic VIPAs 218
Configuring the Multiple Application-Instance Scenario 219
Configuring the Unique Application-Instance Scenario 219
Choosing which form of Dynamic VIPA support to use 223
Configuring Distributed DVIPAs — Sysplex Distributor 224
Sysplex wide source VIPA 226
Sysplex Wide Security Associations 228
Resolution of Dynamic VIPA conflicts 232
Restart of the original VIPADEFINE TCP/IP after an outage 232
Movement of unique application-instance (BIND) 234
Movement of a unique APF-authorized application instance (ioctl) 235
Same Dynamic VIPA as VIPADEFINE and BIND(), SIOCSVIPA ioctl, or MODDVIPA utility 235
Dynamic VIPA creation results 236
Other considerations 239
Mixture of types of Dynamic VIPAs within subnets 239

MVS failure and Sysplex Failure Management . . . 239
Applications and Dynamic VIPAs . . . 239
Example of configuring Dynamic and Distributed VIPAs . . . 240
Verifying the DVIPAs in a sysplex . . . 241
Using NETSTAT support to verify Dynamic VIPA configuration . . . 244
Verifying Sysplex Distributor workload . . . 246
Dynamic VIPAs and routing protocols . . . 247
OMPROUTE . . . 247
RIP (Routing Information Protocol) . . . 249

Chapter 6. TCP/IP in a sysplex . . . 251
Connectivity in a sysplex . . . 252
Dynamic XCF . . . 252
Workload balancing . . . 260
Single systemwide image . . . 260
Horizontal growth . . . 260
Ease of management . . . 260
DNS/WLM . . . 261
External IP workload balancing . . . 261
Sysplex Distributor . . . 261

Part 2. Server applications . . . 267

Chapter 7. Network connectivity with an SNA network . . . 269
SNALINK LU0 environment . . . 269
Understanding the SNALINK environment . . . 269
Configuring SNALINK LU0 . . . 270
Stopping and starting SNALINK . . . 273
Verifying connection status using NETSTAT DEVLINKS . . . 275
Controlling the SNALINK LU0 interface with the MODIFY command . . . 275
SNALINK LU6.2 . . . 276
Configuring SNALINK LU6.2 . . . 276
Sample console . . . 278
X.25 NCP Packet Switching Interface (NPSI) . . . 278
Configuring X.25 NPSI . . . 279
NCPROUTE . . . 285
Understanding the NCPROUTE environment . . . 286
Configuring NCPROUTE . . . 290

Chapter 8. Accessing remote hosts using Telnet . . . 305
TN3270 Telnet server . . . 305
Getting started . . . 306
Managing the Telnet server . . . 309
Connection mode choices . . . 314
Connection security . . . 319
Mapping Objects to Client Identifiers . . . 325
Mapping methods . . . 335
Advanced LU mapping topics . . . 344
Advanced application topics . . . 354
Device types and logmode considerations . . . 361
Using the Telnet Solicitor or USS logon panel . . . 362
Timers . . . 366
Telnet diagnostics . . . 367
WorkLoad Manager for Telnet (WLM) . . . 372
Configuring the z/OS UNIX Telnet server (otelnetd) . . . 374
Installation information . . . 374

Starting, stopping, and administration of z/OS UNIX Telnet 375
otelnetd 378
SMF record handling 381
BPX.DAEMON considerations 381
Kerberos. 381

Chapter 9. Transferring files using FTP 383
Configuring PROFILE.TCPIP for FTP 384
Configuring ETC.SERVICES 385
Configuring /etc/syslog.conf 385
Configuring the FTPD cataloged procedure 385
Security considerations for the FTP server 386
Defining environment variables for the FTP server (optional). 388
Configuring TCPIP.DATA for FTP 389
Configuring FTP.DATA. 389
Optionally configuring user-level server options using FTPS.RC 390
Data set attributes 390
Specifying attributes for new MVS data sets. 391
Translation of data 393
Accounting 393
Configure the FTP server for SMF (optional) 393
Customizing the FTP server for TLS 394
Customizing the FTP server for the GSSAPI 395
DB2® and JES 396
Configuring the optional FTP user exits 396
The FTPSMFEX user exit 396
The FTCHKIP user exit 396
The FTCHKPWD user exit 397
The FTCHKCMD user exit 397
The FTCHKJES user exit 398
The FTPOSTPR user exit 398
Customizing the FTP-to-JES interface for JESINTERFACELevel 2 (optional) 399
Configuring the FTP server for anonymous logins (optional) 400
Creating an anonymous directory structure in the HFS 402
Configure the Welcome Banner Page, Login, and Directory Message (optional) 405
Using magic cookies to represent information 405
Configuring to send detailed login failure replies to an FTP client (optional) 406
Install the SQL query function (optional) and access the DB2 modules 406
Accessing DB2 modules 408
FTP.DATA updates for SQL query function 408
Trivial File Transfer Protocol (TFTP). 408
Considerations for z/OS 408
Verification of FTP 410
Verify server 410
Verify client 411
Verify FTP.DATA statements 412
Verifying anonymous, banner, and other optional configuration information 414
Verify FTP-JES interface (optional) 414

Chapter 10. Domain Name System (DNS) 417
DNS and BIND overview 417
Domain names 418
Domain name servers 419
Resolvers 422
Recommended reading 424
Migrating to BIND 9. 424

Performance issues. 424
Compatibility considerations. 425
Zone transfers 425
Queries 425
Dynamic update 426
DNSSEC 426
TSIG 426
DNS/WLM (Sysplex connection balancing) 426
IPv6 support 426
Stack affinity 426
NOTIFY 426
Running the name server in BIND 9 and BIND 4.9.3 mode simultaneously 426
Setting up and running the name server 427
Configuring a master (primary) name server. 427
Configuring a slave name server 450
Configuring a cache-only name server 453
Configuring a stealth name server 456
Adding forwarding to your name server 456
Configuring host resolvers: Name server considerations 457
Configuring host resolvers: onslookup considerations 457
Creating the syslog file 458
BIND 9 security considerations 458
Special considerations when using Dynamic VIPA 462
Dynamic primary DNS movement using Dynamic VIPA. 462
Querying name servers 463
nslookup command 463
Diagnosing problems 466
Checking messages sent to the operators console 467
Checking the syslog messages 467
Using name server signals to diagnose BIND 4.9.3 DNS problems 467
Using name server signals to diagnose BIND 9 DNS problems. 467
Using rndc to diagnose BIND 9 problems. 468
Checking name server logging files to diagnose BIND 9 468
Using nslookup to diagnose problems 468
Using dig to diagnose problems 469
Advanced BIND 9 name server topics 469
Multiple TCP/IP stack (common INET) considerations 469
Dynamic update 470
Incremental zone transfers (IXFR) 470
Split DNS 471
TSIG 475
DNSSEC 476
IPv6 support in BIND 9 479
Advanced BIND 4.9.3–Name server topics 482
Connection optimization in a sysplex domain 482
Dynamic IP 496
DNS-related RFCs 535
Proposed standards 535
Proposed standards still under development 535
Other important RFCs about DNS implementation 536
Resource record types 536
DNS and the Internet 536
DNS operations 536
Other DNS-related RFCs. 536

Chapter 11. Policy-Based Networking 539

The role of policy . . . 539
Policy components overview . . . 539
Policy Agent . . . 539
RSVP Agent . . . 540
SNMP SLA subagent . . . 540
Intrusion Detection Services . . . 540
Policy sample files . . . 541
Policy object model overview . . . 543
What kind of policy do you want? . . . 546
QoS policy . . . 546
IDS policy . . . 547
Where do you want to define your policies? . . . 547
LDAP server . . . 548
Overview of the object classes . . . 548
Considerations for defining LDAP objects . . . 554
Policy Agent retrieval of LDAP objects . . . 554
Installing the schema definition on the LDAP server . . . 555
Using the sample LDAP objects . . . 556
Policy Agent common functions . . . 557
Configuring the Policy Agent . . . 557
Starting and stopping the Policy Agent . . . 562
Refreshing policies . . . 563
Verification . . . 563
Are the policies defined correctly to the LDAP server? . . . 563
Are the policies defined correctly to the Policy Agent? . . . 563

Chapter 12. Quality of Service (QoS) . . . 565
Differentiated Services (DS) policies . . . 565
Integrated Services (RSVP) policies . . . 567
Sysplex Distributor (SD) policies . . . 567
QoS specific Policy Agent functions . . . 567
Sysplex distributor policy performance monitoring configuration . . . 568
Type of Service (ToS) mapping configuration . . . 569
Defining policies in a Policy Agent configuration file . . . 570
Differentiated Services policy examples . . . 571
RSVP policy example . . . 572
Sysplex Distributor policy example . . . 573
Defining policies using LDAP . . . 574
Differentiated Services policy example . . . 574
RSVP policy example . . . 579
Sysplex Distributor routing policy example . . . 580
RSVP . . . 583
Configuring the RSVP agent . . . 584
Starting and stopping RSVP . . . 584
Service Level Agreement Performance Monitor MIB subagent . . . 585
Starting and stopping the SLA subagent . . . 585
Verification . . . 586
Are the policies installed in the TCP/IP stacks? . . . 586
Is the expected traffic mapping to the correct QoS policies? . . . 586
Are the Sysplex Distributor policy functions working correctly? . . . 587
Does anything need to be tuned? . . . 587
Using PASEARCH . . . 587
Using the SLA subagent to monitor policies . . . 588

Chapter 13. Intrusion Detection Services (IDS) . . . 595
Scan policies . . . 598

Attack policies 598
Traffic Regulation (TR) policies 601
TR TCP 601
TR UDP 602
Defining TR TCP policies using the Policy Agent 603
Defining IDS policies using LDAP 603
IDS policy definition considerations 603
IDS scan policy example 605
IDS attack policy examples 608
Traffic Regulation (TR) policy examples 615
Verification 619
Are the correct policies active? 619
Is the expected traffic mapping to the correct policies? 619
Are the IDS policy functions working correctly? 619
TRMD. 620
Running TRMD as a started task 620
Running TRMD from the z/OS UNIX shell 620
Stopping TRMD 621
TRMDSTAT. 621

Chapter 14. Network management 623
Overview of SNMP 623
Overview of z/OS CS SNMP version 3. 624
Processing an SNMP request 624
Deciding on SNMP security needs 625
Step 1: Configure the SNMP agent (OSNMPD) 627
Provide TCP/IP profile statements 627
Provide community-based security and notification destination information 629
Provide community-based and user-based security and notification destination information 631
Provide security product access to agent from subagents 634
Provide MIB object configuration information 634
Start the SNMP agent (OSNMPD) 635
Sample JCL procedure for starting OSNMPD from MVS 636
Starting OSNMPD from z/OS UNIX 636
Step 2: Configure the SNMP commands 636
Configure the NetView SNMP (SNMP) command 637
Configure the z/OS UNIX SNMP (osnmp) command. 640
Step 3: Configure the SNMP subagents 642
Step 4: Configure the Open Systems Adapter (OSA) support 642
OSA/SF prerequisites 644
Required TCP/IP profile statements 645
Multiple TCP/IP instances considerations 645
Step 5: Configure the trap forwarder daemon 646
Provide PROFILE.TCPIP statements 647
Provide trap forwarder configuration information 647
Starting and stopping the trap forwarder daemon 647

Chapter 15. Remote Print Server (LPD). 649
Configuring the Remote Print Server 649
Step 1: Configuring PROFILE.TCPIP for LPD 649
Step 2: Updating the LPD server cataloged procedure 650
Step 3: Updating the LPD server configuration data set 651
Step 4: Creating a banner page (optional) 651

Chapter 16. Remote procedure calls. 653

Configuring the PORTMAP address space 653
Step 1: Configuring PROFILE.TCPIP for PORTMAP. 653
Step 2: Updating the PORTMAP cataloged procedure 654
Step 3: Defining the data set for well-known procedure names 654
Starting the PORTMAP address space. 656
Configuring the z/OS UNIX PORTMAP address space 656
Step 1: Configuring PROFILE.TCPIP for UNIX PORTMAP 656
Step 2: Updating the PORTMAP cataloged procedure 657
Starting the PORTMAP address space. 657
Configuring the NCS interface 657
Understanding the LLBD server 658
Understanding the NRGLBD server 658
Step 1: Configuring PROFILE.TCPIP for NCS 658
Step 2: Updating the NRGLBD cataloged procedure. 659
Step 3: Updating the LLBD cataloged procedure 659
Configuring the Network Database (NDB) System 659
Step 1: Updating the NDB setup sample job 660
Step 2: Running the NDB setup job 660
Step 3: Updating and installing the DB2 sample connection exit routine 660
Step 4: Updating the PORTS cataloged procedure 662
Step 5: Updating the PORTC cataloged procedure 662
Step 6: Creating the NDB clients 662
Starting NDB 668

Chapter 17. Mail servers 669
Configuring the SMTP server 669
Checklist for working within the SMTP environment 669
Configuration process 670
Configuring z/OS UNIX sendmail and popper 689
Overview 689
Configuring z/OS UNIX sendmail 691
sendmail as a daemon 696
Configuring popper 696

Chapter 18. TIMED daemon 699
Starting TIMED from z/OS shell 699
Starting TIMED as a procedure 699

Chapter 19. SNTPD daemon 701
Steps for starting SNTPD from the z/OS shell 701
Steps for starting SNTPD as a procedure. 702
Stack affinity 703

Chapter 20. Remote Execution 705
UNIX REXEC 705
TSO REXEC 705
Configuring the TSO Remote Execution server. 705
Step 1: Configuring PROFILE.TCPIP for TSO Remote Execution server 705
Step 2: Determine whether Remote Execution client will send REXEC or RSH commands 706
Step 3: Permit remote users to access MVS resources (optional) 706
Step 4: Update the TSO Remote Execution cataloged procedure 707
Step 5: Create a user exit routine (optional) 707
Configuring the z/OS UNIX Remote Execution servers 708
Installation information. 708

Configuring TSO and z/OS UNIX Remote Execution servers to use the same port . . . 710

Chapter 21. Miscellaneous (MISC) server . . . 713
Discard protocol . . . 713
Echo protocol . . . 713
Character generator protocol . . . 713
Configuring the MISC server . . . 714
Step 1: Configuring PROFILE.TCPIP for the MISC server . . . 714
Step 2: Updating the MISC server cataloged procedure (MISCSERV) . . . 715

Part 3. Appendixes . . . 717

Appendix A. Setting up the inetd configuration file . . . 719

Appendix B. TLS/SSL security . . . 721
Secure Socket Layer overview . . . 721
Server authentication . . . 722
Client authentication . . . 724
Encryption algorithms . . . 724
Creating and managing keys and certificates at the server . . . 726
Overview . . . 726
Using the gskkyman utility . . . 728
Using RACF's common keyring support . . . 734
Migrating an existing gskkyman key database to RACF . . . 739
Creating and managing keys and certificates at the client . . . 740
Create a self-signed client certificate . . . 740
Add server certificates to the client keyring . . . 744

Appendix C. Express Logon Feature (ELF) . . . 749
Configuring RACF services for Express Logon . . . 750
Configuring the Express Logon components . . . 751
Configuring the HOD V5 TN3270 client . . . 751
Configuring the z/OS TN3270 server . . . 752
Configuring the middle-tier TN3270 server (CS/2 example) . . . 752
Configuring the Digital Certificate Access Server (DCAS) . . . 752

Appendix D. Using HCD . . . 757

Appendix E. Configuring the OROUTED server . . . 769
Understanding OROUTED . . . 769
Routing Information Protocol (RIP) . . . 770
RIP Version 2 . . . 771
OROUTED miscellaneous features . . . 771
RIP input/output filters . . . 772
RIP routes . . . 772
OROUTED gateways . . . 773
Passive RIP routes . . . 773
External RIP routes . . . 773
Active gateways . . . 773
OROUTED gateways summary . . . 774
OROUTED configuration process . . . 774
Step 1: Configure the OROUTED profile . . . 774
Step 2: Update configuration statements in PROFILE.TCPIP . . . 777
Step 3: Update the resolver configuration file . . . 778
Step 4: Update the OROUTED cataloged procedure (optional) . . . 779

OROUTED cataloged procedure 779
Step 5: Specify the OROUTED port number in the SERVICES file 779
Step 6: Configure the gateways file or data set (optional) 779
Syntax rules 780
Step 7: Configure and start syslogd 785
Step 8: RACF-authorize user IDs 785
OROUTED parameters 786
Specifying parameters 788
Starting OROUTED 788
Configuration examples 789
Configuring a passive route 789
Configuring an external route 790
Configuring an active gateway 791
Configuring a point-to-point link 792
Configuring a default route 792
Configuring ORouteD with Enterprise Extender 792
Configuring OROUTED with VIPA 793
Configuring OROUTED to split traffic with VIPA 793
Configuring OROUTED with OSA-Express in QDIO mode 795
Configuring OROUTED with HiperSockets 796

Appendix F. Related protocol specifications (RFCs) 797
Draft RFCs 804

Appendix G. Information APARs 807
Information APARs for IP documents 807
Information APARs for SNA documents 808
Other information APARs 808

Appendix H. Accessibility 811
Using assistive technologies 811
Keyboard navigation of the user interface 811

Notices 813
Trademarks 816

Index 819

Communicating Your Comments to IBM 833

Figures

1. Resolver related configuration files in z/OS UNIX and native MVS environments . . . 27
2. syslogd operation . . . 39
3. Generic server . . . 56
4. Server with affinity for a specific transport provider . . . 57
5. Example of binding an application to a specific transport provider . . . 57
6. REXX program to switch TSO user to another TCP/IP stack . . . 63
7. SYS1.PARMLIB(BPXPRMxx) for CINET . . . 64
8. Syntax for TCP/IP message IDs . . . 77
9. Elements of a secure TCP/IP deployment . . . 79
10. User identification, authentication, and access control for z/OS Communications Server applications . . . 80
11. Stack Access Control overview . . . 82
12. Port Access Control overview . . . 83
13. Network Access Control example . . . 85
14. IP filtering at the z/OS communication endpoint . . . 87
15. Security protocols from a protocol layering perspective . . . 88
16. e-business scenarios with Virtual Private Networks . . . 89
17. IPSec AH protocol header formats and security coverage . . . 90
18. IPSec ESP protocol header formats and security coverage . . . 91
19. IPSec and IKE overview . . . 92
20. TN3270 SSL overview . . . 94
21. Using multiple TN3270 ports to separate SSL and non-SSL traffic . . . 95
22. Combining TN3270 SSL with IPSec client-to-firewall authentication . . . 95
23. TN3270 SSL and non-SSL traffic using a single TN3270 port . . . 96
24. FTP client and server TLS overview . . . 97
25. Intrusion Detection Services overview . . . 100
26. Example of TCP/IP operating characteristics in PROFILE.TCPIP . . . 112
27. Example of physical characteristics in PROFILE.TCPIP . . . 115
28. HiperSockets Virtual LAN . . . 130
29. HiperSockets multiple LANs . . . 131
30. Candidate configuration for HiperSockets Accelerator . . . 135
31. HiperSockets Accelerator configuration . . . 136
32. Example of reserved port number definitions . . . 139
33. Sample network . . . 158
34. Static VIPA configuration . . . 214
35. Sample DVIPA addressing in a sysplex environment . . . 217
36. DVIPA takeover with SWSA . . . 230
37. Sysplex Distributor with SWSA . . . 231
38. SNALINK environment interfaces . . . 270
39. SNA DLC link . . . 271
40. APPL statement for SNALINK . . . 273
41. SNALINK console example . . . 274
42. APPL statement for SNALINK LU6.2 . . . 277
43. Sample MVS system console messages on SNALINK LU6.2 address space startup . . . 278
44. NCPROUTE environment . . . 286
45. NCPROUTE example configuration . . . 291
46. NCPROUTE data sets relationship . . . 300
47. NCPROUTE configuration example of a passive route . . . 301
48. Configuring an active gateway . . . 302
49. Telnet connectivity . . . 305
50. Telnet parameter placement . . . 308
51. Telnet profiles and connections . . . 313
52. Port 1023 connection characteristics . . . 324

53. Mapping model 325
54. Search method 333
55. Session initiation failures scenarios 355
56. Session ending scenarios 356
57. Hierarchical naming tree 419
58. Name resolution to a sysplex 484
59. Address association with mvsplex.mycorp.com 486
60. Address association with myserver 487
61. Policy components in z/OS CS 541
62. Basic policy objects 543
63. Complex policy conditions 544
64. Rule-specific conditions and actions 545
65. Reusable conditions and actions 545
66. Policy groups. 546
67. LDAP schema object class hierarchy 551
68. Overview of SNMP support 624
69. Configuration files for SNMP agent. 627
70. Configuration files for NetView SNMP. 637
71. Configuration files for osnmp 640
72. Subagent connection to OSA/SF 646
73. Sender MUA transmits the message to sendmail 689
74. sendmail transmits the message to an intermediate SMTP server 690
75. A sendmail daemon receives the message from an SMTP client 690
76. sendmail delivers the message to the local recipient 690
77. Receiver's MUA has direct access to the mail spool file 691
78. Receiver's MUA retrieves the message over a POP3 connection with a popper daemon 691
79. Adding applications to /etc/inetd.conf 719
80. Setting traces in /etc/inetd.conf 719
81. IBM Keys Management 741
82. Create New Self-Signed Certificate. 741
83. IBM Key Management 742
84. Export/Import Key 742
85. Extract Certificate to a File. 743
86. HOD connection using a client certificate 743
87. HOD security properties. 744
88. Security Information 745
89. Extract a Certificate 745
90. Certificate was extracted 745
91. Creating a new CustomizedCAs.class. 746
92. Default location displayed 746
93. Add CA's Certificate From a File 746
94. Add CA's Certificate From a File — continued 747
95. Express Logon network 749
96. Select processors 757
97. Work with attached channel paths 757
98. Initiate the Define Channel Path dialog 758
99. Add channel path 758
100. Specify Maximum Frame Size 759
101. Define the channel path access list 760
102. Channel path number FF defined 760
103. Work with attached control units. 761
104. Add the control unit(s) 761
105. Define a control unit 762
106. Define it to the processor 762
107. Currently defined control unit 763
108. Define the devices. 763

109. Empty device list 764
110. Define the devices for the control unit. 764
111. Add devices of type IQD 765
112. Define number of devices 765
113. Define device to operating system 766
114. Select systems 766
115. Complete the definition 767
116. Definition completed 767
117. Sample OROUTED configuration file 777
118. Sample portion of services file 779
119. Example commands to start multiple copies of OROUTED 789
120. OROUTED configuration example 790
121. Configuring an active gateway 791
122. Single VIPA configuration 794
123. Multiple VIPA configuration. 795

Tables

| 1. z/OS TCP/IP stack function support . . . 3
2. TCP/IP configuration data sets . . . 21
| 3. Local definitions available to resolver . . . 27
4. syslogd facilities . . . 40
5. Setting up default of OMVS segment . . . 46
6. BPX.DAEMON . . . 48
7. Program control . . . 49
8. How your own socket programs select a stack . . . 61
9. Interior Gateway Protocol characteristics . . . 156
10. Multipath route limitations . . . 171
11. Route precedence . . . 192
12. Summary of Dynamic VIPA creation results . . . 236
13. RIP route advertising rules . . . 288
14. NCPROUTE gateways summary . . . 290
| 15. Client 1 example . . . 333
| 16. Client 2 example . . . 334
| 17. Client 3 example . . . 335
18. PORTCOMMAND scenarios . . . 387
19. Settings that affect nslookup operation . . . 465
20. DHCP server configuration . . . 497
21. Monitor control and monitor status object bit values . . . 591
22. Security advantages and disadvantages . . . 626
23. Summary of SMTP configuration statements . . . 682
24. Required and recommended m4 items . . . 693
25. Sendmail permission table . . . 695
26. Frame size specification . . . 759
27. OROUTED gateways summary . . . 774
28. ORouteD parameters . . . 788
29. IP information APARs . . . 807
30. SNA information APARs . . . 808
31. Non-document information APARs . . . 809

About this document

This document contains guidance material to enable you to configure IP address spaces, servers, and applications for z/OS™ Communications Server. This volume is part of a two-volume set:

- *z/OS Communications Server: IP Configuration Guide*, which contains concepts and guidance, explaining an overall approach to IP configuration.
- *z/OS Communications Server: IP Configuration Reference*, which describes parameters and options, and syntax of statements.

The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

For detailed information about configuration-related data sets and statements, refer to *z/OS Communications Server: IP Configuration Reference*.

For detailed information about commands used during configuration, refer to *z/OS Communications Server: IP System Administrator's Commands*.

This document supports z/OS.e™.

Who should use this document

This document is intended for programmers and system administrators who are familiar with TCP/IP, MVS™, z/OS UNIX®, and the Time Sharing Option Extensions (TSO/E).

Where to find more information

This section contains:

- Pointers to information available on the Internet
- Information about licensed documentation
- Information about LookAt, the online message tool
- A set of tables that describes the documents in the z/OS Communications Server (z/OS CS) library, along with related publications

Where to find related information on the Internet

z/OS

– http://www.ibm.com/servers/eserver/zseries/zos/

z/OS Internet Library

– http://www.ibm.com/servers/eserver/zseries/zos/bkserv/

IBM Communications Server product

– http://www.software.ibm.com/network/commserver/

IBM Communications Server product support

– http://www.software.ibm.com/network/commserver/support/

IBM Systems Center publications

– http://www.redbooks.ibm.com/

IBM Systems Center flashes

– http://www-1.ibm.com/support/techdocs/atsmastr.nsf

RFCs

– http://www.ietf.org/rfc.html

RFC drafts

– http://www.ietf.org/ID.html

Information about Web addresses can also be found in information APAR II11334.

DNS web sites

For more information about DNS, see the following USENET news groups and mailing:

USENET news groups:

comp.protocols.dns.bind

For BIND mailing lists, see:

- http://www.isc.org/ml-archives/
  - BIND Users
    - Subscribe by sending mail to bind-users-request@isc.org.
    - Submit questions or answers to this forum by sending mail to bind-users@isc.org.
  - BIND 9 Users (Note: This list may not be maintained indefinitely.)
    - Subscribe by sending mail to bind9-users-request@isc.org.
    - Submit questions or answers to this forum by sending mail to bind9-users@isc.org.

For definitions of the terms and abbreviations used in this document, you can view or download the latest *IBM Glossary of Computing Terms* at the following Web address:

http://www.ibm.com/ibm/terminology

Note: Any pointers in this publication to Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

`http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/`

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm Pilot VIIx suggested).

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the **News** section on the LookAt Web site.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

How to contact IBM service

For immediate assistance, visit this Web site:
http://www.software.ibm.com/network/commserver/support/

Most problems can be resolved at this Web site, where you can submit questions and problem reports electronically, as well as access a variety of diagnosis information.

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-237-5511). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside of the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

If you would like to provide feedback on this publication, see “Communicating Your Comments to IBM” on page 833.

z/OS Communications Server information

This section contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server publications are available:

- Online at the z/OS Internet Library web page at
  http://www.ibm.com/servers/eserver/zseries/zos/bkserv

- In softcopy on CD-ROM collections.

Softcopy information

Softcopy publications are available in the following collections:

Titles	Order Number	Description
z/OS V1R4 Collection	SK3T-4269	This is the CD collection shipped with the z/OS product. It includes the libraries for z/OS V1R4, in both BookManager® and PDF formats.
z/OS Software Products Collection	SK3T-4270	This CD includes, in both BookManager and PDF formats, the libraries of z/OS software products that run on z/OS but are not elements and features, as well as the *Getting Started with Parallel Sysplex®* bookshelf.
z/OS V1R4 and Software Products DVD Collection	SK3T-4271	This collection includes the libraries of z/OS (the element and feature libraries) and the libraries for z/OS software products in both BookManager and PDF format. This collection combines SK3T-4269 and SK3T-4270.
z/OS Licensed Product Library	SK3T-4307	This CD includes the licensed documents in both BookManager and PDF format.
System Center Publication IBM S/390® Redbooks™ Collection	SK2T-2177	This collection contains over 300 ITSO redbooks that apply to the S/390 platform and to host networking arranged into subject bookshelves.

z/OS Communications Server library

z/OS V1R4 Communications Server documents are available on the CD-ROM accompanying z/OS (SK3T-4269 or SK3T-4307). Unlicensed documents can be viewed at the z/OS Internet library site.

Updates to documents are available on RETAIN® and in information APARs (info APARs). See Appendix G, “Information APARs” on page 807 for a list of the documents and the info APARs associated with them.

- Info APARs for OS/390® documents are in the document called *OS/390 DOC APAR and PTF ++HOLD Documentation* which can be found at http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IDDOCMST/CCONTENTS.
- Info APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation* which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

Planning and migration:

Title	Number	Description
z/OS Communications Server: SNA Migration	GC31-8774	This document is intended to help you plan for SNA, whether you are migrating from a previous version or installing SNA for the first time. This document also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with SNA.
z/OS Communications Server: IP Migration	GC31-8773	This document is intended to help you plan for TCP/IP Services, whether you are migrating from a previous version or installing IP for the first time. This document also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with TCP/IP Services.

Title	Number	Description
z/OS Communications Server: IPv6 Network and Application Design Guide	SC31-8885	This document is a high-level introduction to IPv6. It describes concepts of z/OS Communications Server's support of IPv6, coexistence with IPv4, and migration issues.

Resource definition, configuration, and tuning:

Title	Number	Description
z/OS Communications Server: IP Configuration Guide	SC31-8775	This document describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this document in conjunction with the *z/OS Communications Server: IP Configuration Reference*.
z/OS Communications Server: IP Configuration Reference	SC31-8776	This document presents information for people who want to administer and maintain IP. Use this document in conjunction with the *z/OS Communications Server: IP Configuration Guide*. The information in this document includes: • TCP/IP configuration data sets • Configuration statements • Translation tables • SMF records • Protocol number and port assignments
z/OS Communications Server: SNA Network Implementation Guide	SC31-8777	This document presents the major concepts involved in implementing an SNA network. Use this document in conjunction with the *z/OS Communications Server: SNA Resource Definition Reference*.
z/OS Communications Server: SNA Resource Definition Reference	SC31-8778	This document describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this document in conjunction with the *z/OS Communications Server: SNA Network Implementation Guide*.
z/OS Communications Server: SNA Resource Definition Samples	SC31-8836	This document contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.
z/OS Communications Server: AnyNet SNA over TCP/IP	SC31-8832	This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.
z/OS Communications Server: AnyNet Sockets over SNA	SC31-8831	This guide provides information to help you install, configure, use, and diagnose sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.
z/OS Communications Server: IP Network Print Facility	SC31-8833	This document is for system programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services.

Operation:

Title	Number	Description
z/OS Communications Server: IP User's Guide and Commands	SC31-8780	This document describes how to use TCP/IP applications. It contains requests that allow a user to log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.
z/OS Communications Server: IP System Administrator's Commands	SC31-8781	This document describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator's commands, such as TSO NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS commands commonly used during the IP configuration process.
z/OS Communications Server: SNA Operation	SC31-8779	This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.
z/OS Communications Server: Quick Reference	SX75-0124	This document contains essential information about SNA and IP commands.

Customization:

Title	Number	Description
z/OS Communications Server: SNA Customization	LY43-0092	This document enables you to customize SNA, and includes the following: • Communication network management (CNM) routing table • Logon-interpret routine requirements • Logon manager installation-wide exit routine for the CLU search exit • TSO/SNA installation-wide exit routines • SNA installation-wide exit routines

Writing application programs:

Title	Number	Description
z/OS Communications Server: IP Application Programming Interface Guide	SC31-8788	This document describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this document to adapt your existing applications to communicate with each other using sockets over TCP/IP.
z/OS Communications Server: IP CICS Sockets Guide	SC31-8807	This document is for programmers who want to set up, write application programs for, and diagnose problems with the socket interface for CICS® using z/OS TCP/IP.
z/OS Communications Server: IP IMS Sockets Guide	SC31-8830	This document is for programmers who want application programs that use the IMS™ TCP/IP application development services provided by IBM's TCP/IP Services.

Title	Number	Description
z/OS Communications Server: IP Programmer's Reference	SC31-8787	This document describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.
z/OS Communications Server: SNA Programming	SC31-8829	This document describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.
z/OS Communications Server: SNA Programmer's LU 6.2 Guide	SC31-8811	This document describes how to use the SNA LU 6.2 application programming interface for host application programs. This document applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this document.)
z/OS Communications Server: SNA Programmer's LU 6.2 Reference	SC31-8810	This document provides reference material for the SNA LU 6.2 programming interface for host application programs.
z/OS Communications Server: CSM Guide	SC31-8808	This document describes how applications use the communications storage manager.
z/OS Communications Server: CMIP Services and Topology Agent Guide	SC31-8828	This document describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The document provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis:

Title	Number	Description
z/OS Communications Server: IP Diagnosis	GC31-8782	This document explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.
z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures and *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT*	LY43-0088 LY43-0089	These documents help you identify an SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.
z/OS Communications Server: SNA Data Areas Volume 1 and *z/OS Communications Server: SNA Data Areas Volume 2*	LY43-0090 LY43-0091	These documents describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and codes:

Title	Number	Description
z/OS Communications Server: SNA Messages	SC31-8790	This document describes the ELM, IKT, IST, ISU, IUT, IVT, and USS messages. Other information in this document includes: • Command and RU types in SNA messages • Node and ID types in SNA messages • Supplemental message-related information
z/OS Communications Server: IP Messages Volume 1 (EZA)	SC31-8783	This volume contains TCP/IP messages beginning with EZA.
z/OS Communications Server: IP Messages Volume 2 (EZB)	SC31-8784	This volume contains TCP/IP messages beginning with EZB.
z/OS Communications Server: IP Messages Volume 3 (EZY)	SC31-8785	This volume contains TCP/IP messages beginning with EZY.
z/OS Communications Server: IP Messages Volume 4 (EZZ-SNM)	SC31-8786	This volume contains TCP/IP messages beginning with EZZ and SNM.
z/OS Communications Server: IP and SNA Codes	SC31-8791	This document describes codes and other information that appear in z/OS Communications Server messages.

APPC Application Suite:

Title	Number	Description
z/OS Communications Server: APPC Application Suite User's Guide	SC31-8809	This documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.
z/OS Communications Server: APPC Application Suite Administration	SC31-8835	This document contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.
z/OS Communications Server: APPC Application Suite Programming	SC31-8834	This document provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Redbooks

The following Redbooks may help you as you implement z/OS Communications Server.

Title	Number
TCP/IP Tutorial and Technical Overview	GG24–3376
SNA and TCP/IP Integration	SG24–5291
IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 1: Configuration and Routing	SG24–5227
IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide: Volume 2: UNIX Applications	SG24–5228
IBM Communications Server for OS/390 V2R7 TCP/IP Implementation Guide: Volume 3: MVS Applications	SG24–5229
Secureway Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements	SG24–5631
TCP/IP in a Sysplex	SG24–5235
Managing OS/390 TCP/IP with SNMP	SG24–5866

Title	Number
Security in OS/390–based TCP/IP Networks	SG24–5383
IP Network Design Guide	SG24–2580
Migrating Subarea Networks to an IP Infrastructure	SG24–5957
IBM Communication Controller Migration Guide	SG24–6298

Related information

For information about z/OS products, refer to *z/OS Information Roadmap* (SA22-7500). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, as well as describing each z/OS publication.

Relevant RFCs are listed in an appendix of the IP documents. Architectural specifications for the SNA protocol are listed in an appendix of the SNA documents.

The table below lists documents that may be helpful to readers.

Title	Number
z/OS Security Server Firewall Technologies	SC24-5922
S/390: OSA-Express Customer's Guide and Reference	SA22-7403
z/OS JES2 Initialization and Tuning Guide	SA22-7532
z/OS MVS Diagnosis: Procedures	GA22-7587
z/OS MVS Diagnosis: Reference	GA22-7588
z/OS MVS Diagnosis: Tools and Service Aids	GA22-7589
z/OS Security Server LDAP Client Programming	SC24-5924
z/OS Security Server LDAP Server Administration and Use	SC24-5923
Understanding LDAP	SG24-4986
z/OS UNIX System Services Programming: Assembler Callable Services Reference	SA22-7803
z/OS UNIX System Services Command Reference	SA22-7802
z/OS UNIX System Services User's Guide	SA22-7801
z/OS UNIX System Services Planning	GA22-7800
z/OS MVS Using the Subsystem Interface	SA22-7642
z/OS C/C++ Run-Time Library Reference	SA22-7821
z/OS Program Directory	GI10-0670
DNS and BIND, Fourth Edition, O'Reilly and Associates, 2001	ISBN 0-596-00158-4
Routing in the Internet , Christian Huitema (Prentice Hall PTR, 1995)	ISBN 0-13-132192-7
sendmail, Bryan Costales and Eric Allman, O'Reilly and Associates, 1997	ISBN 156592–222–0
TCP/IP Tutorial and Technical Overview	GG24-3376
TCP/IP Illustrated, Volume I: The Protocols, W. Richard Stevens, Addison-Wesley Publishing, 1994	ISBN 0-201-63346-9
TCP/IP Illustrated, Volume II: The Implementation, Gary R. Wright and W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63354-X
TCP/IP Illustrated, Volume III, W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63495-3
z/OS System Secure Sockets Layer Programming	SC24-5901

Determining if a publication is current

As needed, IBM updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. The following information describes how to determine if you are looking at the most current copy of a publication:

- At the end of a publication's order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
- If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
- To compare softcopy publications, you can check the last two characters of the publication's filename (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

Summary of changes

Summary of changes
for SC31-8775-02
z/OS Version 1 Release 4

This document contains information previously presented in SC31-8775-01, which supports z/OS Version 1 Release 2. The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

New information

- z/OS msys for Setup, see “z/OS msys for Setup and Wizard” on page 6.
- Access control, see “System resource protection” on page 79.
- Transparent fault-tolerance for failed or stopped IPv4 devices or IPv6 interfaces, see “Interface-layer fault-tolerance for local area networks (interface-takeover function)” on page 137.
- Using a single sysplex wide Dynamic VIPA (DVIPA) as the source IP address for TCP applications, and having the sysplex stacks collaborate on assigning ephemeral ports to prevent duplicate connection 4-tuples (combination of source and destination IP addresses and ports), see “Sysplex wide source VIPA” on page 226.
- Sysplex Wide Security Associations (SWSA), see “Sysplex Wide Security Associations” on page 228.
- FTP, see “Translation of data” on page 393 and “Configuring the optional FTP user exits” on page 396.
- Simple Network Time Protocol (SNTP), see Chapter 19, “SNTPD daemon” on page 701.
- The following areas contain new information pertaining to IPv6 support:
  - z/OS TCP/IP stack-related functions and the level of support provided in an IPv6 network, see “z/OS TCP/IP stack function support” on page 3.
  - Sample BPXPRMxx definitions needed for IPv6 support, see “Defining TCP/IP as a UNIX System Services physical file system (PFS)” on page 50.
  - Resolver support, see “Understanding resolvers” on page 12, “Configuration files for TCP/IP applications” on page 26, and “Configuring the local host table (optional)” on page 143.
  - Autoconfiguring addresses for an interface using information provided by IPv6 routers, see “IPv6 considerations: Stateless autoconfiguration and duplicate address detection” on page 137.
  - Routing in an IPv6 network, see Chapter 4, “Routing” on page 155.
  - FTP, see “Security considerations for the FTP server” on page 386.
  - DNS, see Chapter 10, “Domain Name System (DNS)” on page 417.
  - inetd, otelnetd, orexecd, and orshd, see Appendix A, “Setting up the inetd configuration file” on page 719.

An appendix with z/OS product accessibility information has been added.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R4, you may notice changes in the style and structure of some content in this document–for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document supports z/OS.e.

Summary of changes
for SC31-8775-01
z/OS Version 1 Release 2

This document contains information previously presented in SC31-8775-00, which supports z/OS Version 1 Release 1.

New information

- Managed System Infrastructure for Setup, see “z/OS msys for Setup and Wizard” on page 6
- SMF recording enhancements, see “Accounting - SMF records” on page 40
- OSA-Express Token Ring support, see “Setting up physical characteristics in PROFILE.TCPIP” on page 115
- Connection load balancing, see “Connection load balancing using Sysplex Distributor in a network with CISCO routers” on page 264
- New chapter for Security, see Chapter 2, “Security” on page 79
- New chapter for TCP/IP in a Sysplex, see Chapter 6, “TCP/IP in a sysplex” on page 251
- HiperSockets, see “HiperSockets concepts and connectivity” on page 130
- HiperSockets Accelerator, see “Efficient routing using HiperSockets Accelerator” on page 135
- OROUTED to OMPROUTE migration, see “Migration from OROUTED to OMPROUTE” on page 165
- BIND 9-Based DNS, see Chapter 10, “Domain Name System (DNS)” on page 417
- New chapter for Quality of Service, see Chapter 12, “Quality of Service (QoS)” on page 565
- New chapter for Intrusion Detection Services, see Chapter 13, “Intrusion Detection Services (IDS)” on page 595
- SMTP exit to filter unwanted mail, see Chapter 17, “Mail servers” on page 669
- New appendix for SSL/TLS, see Appendix B, “TLS/SSL security” on page 721
- New appendix for Express Logon, see Appendix C, “Express Logon Feature (ELF)” on page 749
- New appendix for using HCD to configure IQD CHPIDs, see Appendix D, “Using HCD” on page 757

Changed information

- Resolver enhancements
- VMCF/TNF sample start procedure (EZAZSSI)
- OMPROUTE to allow RIP1 and RIP2 packets over the same interface
- Replaceable static routes
- OSPF MD5 authentication

- Telnet enhancements
- FTP enhancements
- Netstat enhancements
- Sysplex Distributor policy enhancements
- Policy Agent enhancements
- Application driven policy classification
- Virtual LAN priority tagging

Deleted information

- Kerberos

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes
for SC31-8775-00
z/OS Version 1 Release 1

This document contains information also presented in *OS/390 V2R10 IBM Communications Server: IP Configuration Guide*.

New information

- Added new information in "UNIX System Services security considerations" on page 45. This information was removed from IP Migration.
- Added a new section, "Defining TCP/IP as a UNIX System Services physical file system (PFS)" on page 50. This information was removed from IP Migration.
- Added information to section, "Making SYS1.PARMLIB changes" on page 69. This information was removed from IP Migration.

Part 1. Base TCP/IP system

Chapter 1. Configuration overview

The objective of this chapter is to help you be better prepared for installation related activities. It is important to understand the terms, relationships, and dependencies presented in this chapter as a prerequisite to installation and customization.

After reading this chapter, you will be familiar with:

- z/OS TCP/IP stack function support
- z/OS Managed System Infrastructure for Setup (z/OS msys for Setup)
- z/OS UNIX System Services concepts
- Differences between HFS files and MVS data sets
- Setting up a resolver address space
- Configuration files and their search orders
- MVS Considerations
- Accounting and security issues for the more commonly used daemons
- Defining TCP/IP as a UNIX System Services physical file system (PFS)
- Performance considerations
- Considerations for multiple instances of TCP/IP
- Enterprise Extender considerations
- Virtual IP Address (VIPA) considerations
- The steps required before starting TCP/IP

z/OS TCP/IP stack function support

Table 1 summarizes z/OS TCP/IP stack-related functions and the level of support provided in an IPv6 network. It is anticipated that many more of these functions will be enabled for IPv6 support in subsequent releases of z/OS Communications Server.

In other locations where these functions are described in detail, there might be no statement of support on an IPv6 network. Consult this table to determine whether a given function is applicable to IPv6.

For more information on the related configuration statements for a particular function, refer to *z/OS Communications Server: IP Configuration Reference*.

Table 1. z/OS TCP/IP stack function support

z/OS TCP/IP stack function	IPv4 support?	IPv6 support?	Comments
Link-layer device support			IPv4 devices are defined with the DEVICE and LINK configuration statements. In IPv6, interfaces are defined with the INTERFACE statement.

Table 1. z/OS TCP/IP stack function support (continued)

z/OS TCP/IP stack function	IPv4 support?	IPv6 support?	Comments
Ethernet LAN connectivity using OSA-Express in QDIO mode	Y	Y	To define an MPCIPA device for IPv4, use the DEVICE statement with the MPCIPA parameter and the LINK statement with the IPAQENET parameter. For IPv6 traffic, OSA-Express QDIO fast ethernet and gigabit ethernet support is configured using an INTERFACE statement of type IPAQENET6. This is the only DLC that currently supports IPv6. Related configuration statements: • DEVICE and LINK (MPCIPA devices) • HOME • INTERFACE (IPAQENET6 interfaces)
Virtual IP addressing support			
Virtual device/interface configuration	Y	Y	With IPv4, a static virtual device is configured using DEVICE and LINK statements with the VIRTUAL parameter. An IPv6 virtual interface is configured with an INTERFACE statement of type VIRTUAL6. Related configuration statements: • DEVICE and LINK (VIRTUAL devices) • HOME • INTERFACE (VIRTUAL6 interfaces)
Sysplex support	Y	N	Related configuration statements: • VIPADYNAMIC • IPCONFIG
IP routing functions			
Dynamic routing - OSPF and RIP	Y	N	
Dynamic routing - AutoConfiguration	N	Y	For information on AutoConfiguration, see “IPv6 considerations: Stateless autoconfiguration and duplicate address detection” on page 137.
Static route configuration using BEGINROUTES statement	Y	Y	Related configuration statements: • BEGINROUTES
Static route configuration using GATEWAY statement	Y	N	Related configuration statements: • GATEWAY

Table 1. z/OS TCP/IP stack function support (continued)

z/OS TCP/IP stack function	IPv4 support?	IPv6 support?	Comments
Multipath routing groups	Y	Y	Related configuration statements: • IPCONFIG • IPCONFIG6
Miscellaneous stack functions			
Path MTU discovery	Y	Y	Path MTU discovery is mandatory in IPv6. Related configuration statements: • IPCONFIG • IPCONFIG6
Configurable device or interface recovery interval	Y	Y	Set with the DevRetryDuration keyword on the IPCONFIG statement. Related configuration statements: • IPCONFIG • IPCONFIG6
Link-layer address resolution	Y	Y	In IPv4, performed using Address Resolution Protocol (ARP). In IPv6, performed using neighbor discovery protocol. Related configuration statements: • DEVICE and LINK (LAN Channel Station and OSA devices) • INTERFACE (IPAQENET6 interfaces)
ARP/Neighbor cache PURGE capability	Y	Y	Use the V TCPIP,,PURGECACHE command. For information, see *z/OS Communications Server: IP System Administrator's Commands*.
Datagram forwarding enable/disable	Y	Y	Related configuration statements: • IPCONFIG • IPCONFIG6
Transport-layer functions			
Fast response cache accelerator	Y	N	
Enterprise extender	Y	N	
Server-BIND control	Y	Y	Related configuration statements: • PORT
UDP Checksum disablement option	Y	N	UDP checksum is required when operating over IPv6. Related configuration statements: • UDPCONFIG

Table 1. z/OS TCP/IP stack function support (continued)

z/OS TCP/IP stack function	IPv4 support?	IPv6 support?	Comments
Network management and accounting functions			
SNMP	Y	N	
Policy-based networking	Y	N	
SMF	Y	Y	Type 118 records do not support IPv6 addresses. IPv6 support in type 119 records is being phased in. Currently, only the following records provide IPv6 support: • TCP connection initiation • TCP connection termination • UDP socket close • FTP client transfer completion • FTP server transfer completion • FTP server logon failure Related configuration statements: • SMFCONFIG
Security functions			For an overview of security services, see Chapter 2, “Security” on page 79.
IPSec	Y	N	Related configuration statements: • IPCONFIG
IP filtering	Y	N	
Network access control	Y	N	Related configuration statements: • NETACCESS
Stack and port access control	Y	Y	Related configuration statements: • PORT • DELETE
Intrusion detection services	Y	N	

z/OS msys for Setup and Wizard

Wizard

IBM provides a Web-based wizard called the z/OS IP Configuration Wizard. Use it at any time to configure a single stack, with simple instances of OMPROUTE, FTP, and TN3270 servers, and with all device types, including static VIPA. If you finish using the wizard and complete the tasks defined in the output checklist, you will be ready to use z/OS TCP/IP to communicate with other hosts in your network. The wizard can be found at: http:// www.ibm.com/eserver/zseries/zos/wizards/.

z/OS msys for Setup

A z/OS system is controlled using a multitude of settings, such as parmlib members, /etc files for Unix System Services, or the RACF® database, which are governed by different access methods. There is not a consistent representation of the configuration data; rather system administrators must keep track of various configuration data sets with varying semantics and syntax. While this renders z/OS systems highly flexible, at the same time it makes them complex and laborious to maintain.

Managed System Infrastructure for Setup (Msys for Setup) addresses these difficulties by establishing a central directory for product configuration data and a single interface to this directory. System administrators using msys will configure z/OS by way of GUI panels presented by the msys Windows NT® or Windows® 2000 application. Multiple products can be configured through the same msys application. The configuration data will be collected and stored together as a common entity within an LDAP server. When directed by the system administrator, msys code running on z/OS will extract the configuration data stored in LDAP and produce the various configuration data sets on the z/OS system. This removes the system administrator from the details of the actual configuration statements, parameters and data set locations.

Msys for Setup consists of a collection of GUI panels that run on a Windows NT or Windows 2000 workstation and is connected through IP to an LDAP directory and to an MVS driving system. Msys for Setup provides the infrastructure for an msys exploiter to provide the user the following functionality:

- Refresh/Priming processing - parses a customer's existing configuration files and stores this configuration data, transformed into a tree structure, into an LDAP directory.
- Customization - reads the Refresh data stored in LDAP and populates the data fields on the GUI panels from this Refresh data. The data can then be changed by the system administrator as the panels are navigated. It is also likely that no Refresh step was done, and the panels are presented with either default data or no data. When customization changes are completed, the data is transformed into a tree structure and stored into the LDAP directory.
- Update Processing - after the configuration customization is complete, the administrator can make the 'Perform update' selection. Msys for Setup sends an FTP batch job to the mainframe driving system. Msys for Setup invokes a series of the msys exploiter's Java™ methods that ultimately result in updating msys-created configuration files or creating new configuration files from the new customized data that is extracted from the LDAP directory.
- Commit processing - after update processing is complete, the administrator can make the 'Commit update' selection. If update processing resulted in the creation or modification of temporary msys-created configuration files, those temporary files are copied into more permanent configuration files.

When using TCP/IP's msys for Setup, you will be prompted by GUI panels for customization information, which is then stored in an LDAP directory. Only a subset of TCP/IP's total configuration is supported using msys for Setup.

TCP/IP offers two different levels of service in msys:

- Customization and update processing only
- Refresh, customization, update, and commit processing

Included in the *customization and update processing only* level of service is support for all network devices, routing selections of OSPF or RIP, or use of default routers and static routes. The TN3270 server can be configured in a basic setup, or in advanced mode that supports nearly all TN3270 options. The FTP server is configured to use all defaults. Update processing creates TCP/IP configuration files in a PDS of your choice. The configuration files created are those typically referred to as TCPIP.DATA, PROFILE.TCPIP, and OMPROUTE.CONF, as well as TN3270 and PORTS. Supported configuration statements are listed below:

- For TCPIP.DATA

  DATASETPREFIX
  DOMAINORIGIN
  HOSTNAME
  NSINTERADDR
  TCPIPJOBNAME

  Defaults are used for all other configuration statements.

- For PROFILE.TCPIP

  ATMARPSV
  ATMLIS
  ATMPVC
  AUTOLOG / ENDAUTOLOG
  BEGINROUTES / ENDROUTES and ROUTE
  DEVICE
  HOME
  LINK
  START
  TCPCONFIG RESTRICTLOWPORTS
  TRANSLATE
  UDPCONFIG RESTRICTLOWPORTS

  Defaults are used for all other configuration statements.

- For OMPROUTE

  INTERFACE
  OSPF_INTERFACE
  RIP_INTERFACE

  Defaults are used for all other configuration statements.

- For TN3270

  ALLOWAPPL
  BEGINVTAM / ENDVTAM
  CLIENTAUTH
  CONNTYPE
  DEFAULTAPPL
  DEFAULTLUS
  DEFAULTLUSSPEC
  DEFAULTPRT
  DEFAULTPRTSPEC
  DESTIPGROUP
  DROPASSOCPRINTER
  ENCRYPTION
  EXPRESSLOGON

HNGROUP
INACTIVE
IPGROUP
KEYRING
LINEMODEAPPL
LINKGROUP
LUGROUP
LUMAP
LUSESSIONPEND
MSG07
PORT / SECUREPORT
PRTDEFAULTAPPL
PRTGROUP
PRTMAP
SCANINTERVAL / TIMEMARK
SMFINIT / SMFTERM
SNAEXT
TELNETDEVICE
TELNETGLOBALS / ENDTELNETGLOBALS
TELNETPARMS / ENDTELNETPARMS
TKOSPECLU
USERGROUP
USSTCP

Defaults are used for all other configuration statements.

- For PORTS

  PORT
  PORTRANGE

The *refresh, customization, update, and commit processing* level of service supports only port reservations. Refresh processing is optional, but useful if the administrator's port reservations are in their own configuration file. If no refresh is performed, customization begins with default port reservations. This TCP/IP service can accept port reservations requested by other msys for Setup services such as LDAP. For example, during customization of the LDAP msys for Setup service, the administrator might be asked which port LDAP should use. When the administrator is done customizing the LDAP service, the TCP/IP service would receive a request for that port. Update processing creates or modifies a temporary PORTS configuration file and reserves ports requested by other msys for Setup services. Commit processing copies this temporary file into a more permanent msys PORTS configuration file and can also modify the user's active PROFILE.TCPIP configuration file to use the msys PORTS file. Supported configuration statements are listed below:

- For PORTS

  PORT
  PORTRANGE

For more detailed information on z/OS msys for Setup, refer to *z/OS Managed System Infrastructure for Setup User's Guide*.

z/OS UNIX System Services (z/OS UNIX) concepts

Beginning with MVS/ESA™ Version 4.3 a new type of application program interface was added to the MVS platform with the intent of integrating a UNIX operating system into MVS. Both a C programming API and an interactive environment called the shell were defined to interoperate with UNIX-style files, called Hierarchical File Systems (HFS). Over time, other organizations developed approaches to working with UNIX on various platforms until an organization named X/Open documented standards of what to implement for UNIX interfaces in a series of guides published as the X/Open Portability Guides (XPG). X/Open now owns the term UNIX and certifies different implementations of UNIX according to the UNIX definitions contained in XPG 4.2. In 1996, OS/390 OpenEdition® was awarded UNIX 95 brand certification, thus confirming that it is compliant with all current open industry standards.

Note: In 1998, IBM changed the name OS/390 OpenEdition to OS/390 UNIX System Services.

z/OS UNIX System Services or z/OS UNIX is the z/OS or MVS implementation of UNIX as defined by X/Open in the XPG 4.2. z/OS UNIX coexists with traditional MVS functions and traditional MVS file types (partitioned data sets, sequential files, and so on). It concurrently allows access to HFS files and to UNIX utilities and commands by means of application programming interfaces (APIs) and the interactive SHELL environment. MVS offers two variants of the UNIX SHELL environment:

- The OMVS shell, much like a native UNIX environment
- The ISHELL, an ISPF interface with access to menu-driven command interfaces

With the APIs, programs can run in any environment including batch jobs, in jobs submitted by TSO/E interactive users, and in most other started tasks, or in any other MVS application task environment. The programs can request:

- Only MVS services
- Only z/OS UNIX services
- Both MVS and z/OS UNIX services

The shell interface is an execution environment analogous to TSO/E, with a programming language of shell commands analogous to Restructured eXtended eXecutor (REXX) language. The shell support consists of:

- Programs that are run interactively by shell users
- Shell commands and scripts that are run interactively by shell users
- Shell commands and scripts that are run as batch jobs

Prior to OS/390 V2R5, OS/390 UNIX required APPC/MVS for programs issuing the fork() or spawn() function of OpenEdition callable services. APPC/MVS is no longer required for this purpose. Forked and spawned address spaces are now implemented in z/OS for UNIX processing by the Work Load Manager (WLM) component of MVS.

For a fork(), the system copies one process, called the parent process, into a new process, called the child process, and places the child process in a new address space, the forked address space.

Spawn() also starts a new process in a new address space. Unlike a fork(), in a spawn() call the parent process specifies a name of a program to start the child process.

The types of processes can be:

- User processes, which are associated with a user
- Daemon processes, which perform continuous or periodic functions, such as a Web server

  Daemons are programs that are typically started when the operating system is initialized and remain active to perform standard services. Some programs that initialize processes for users are considered daemons, even though these daemons are not long-running processes. Examples of daemons provided by z/OS UNIX are *cron*, which starts applications at specific times, and *inetd*, which starts applications on demand.

A user or daemon process can have one or more threads. A thread is a single flow of control within a process. Application programmers create multiple threads to structure an application in independent sections that can run in parallel for more efficient use of system resources.

Overview of data sets and HFS files

Data set and *file* are comparable terms. If you are familiar with MVS, you probably use the term data set to describe a unit of data storage. If you are familiar with AIX® or UNIX, you probably use the term file to describe a named set of records stored or processed as a unit. In the TCP/IP environment, in addition to the traditional MVS data set organizations (such as sequential, partitioned) the z/OS UNIX files are arranged in a Hierarchical File System (HFS) and are called HFS files.

Some data sets and HFS files have special importance because of their function. For example, certain data sets and HFS files are used when configuring the TCP/IP environment. Other data sets are used by the Telnet server (Telnet daemon) when performing specific communication functions. See Table 2 on page 21 for descriptions of the data sets and HFS files necessary for configuring the TCP/IP environment and the search orders used to find them. A search order can include both HFS files and data sets, and these data sets and HFS files will be collectively referred to as the *configuration files* in this section.

Note: Not all applications support HFS files.

Hierarchical File System concepts

The Hierarchical File System lets you set up a file hierarchy that consists of:

- HFS **files**, which contain data or programs. A file containing a load module, shell script, or REXX program is called an *executable file*. Files are kept in directories.
- **Directories** that contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside down tree, with root directory at the top and the branches at the bottom. The **root** is the first directory for the file system at the peak of the tree and is designated by a slash (/).
- Additional local or remote **file systems** that are mounted on directories of the root file system or of additional file systems.
- Lastly, the HFS also includes named pipes, links, and other UNIX items. One of these is character special files like /dev/console that are used by applications like

syslogd. Refer to *z/OS UNIX System Services Planning* for more information about UNIX items like character special files.

To the z/OS system, the file hierarchy is a collection of HFS data sets. Each HFS data set is a mountable file system. The root file system is the first file system mounted. Subsequent file systems can be logically mounted on a directory within the root file system or on a directory within any mounted file system.

Except for the direction of the slashes, the Hierarchical File System is similar to a Disk Operating System (DOS) or an OS/2® file system.

Each *mountable* file system resides in an HFS data set on direct access storage. DFSMS/MVS® manages the HFS data sets and the physical files.

The root file system

The root system is the starting point for the overall HFS file structure. It contains the root directory and any related HFS files or subdirectories. The root file system is created as part of the installation process, either the SERVERPAC method or CPBDO, when you install z/OS.

Understanding resolvers

The resolver acts on behalf of programs as a client that accesses name servers for name-to-address or address-to-name resolution. The resolver can also be used to provide protocol and services information. To resolve the query for the requesting program, the resolver can access available name servers, use local definitions (for example, /etc/resolv.conf, /etc/hosts, /etc/ipnodes, HOSTS.SITEINFO, HOSTS.ADDRINFO, or ETC.IPNODES), or use a combination of both. How and if the resolver uses name servers is controlled by TCPIP.DATA statements (resolver directives).

The resolver address space must be started before any application or TCP/IP stack resolver calls can occur. When the resolver address space starts, it reads an optional resolver setup data set pointed to by the SETUP DD card in the resolver JCL procedure. This resolver setup data set enables the following capabilities:

- Specification of a TCPIP.DATA file that contains global settings for the MVS image. The GLOBALTCPIPDATA setup statement identifies the file.

  The global TCPIP.DATA file is being provided to allow the administrator to retain control of which resolver statements are used for name resolution, and to eliminate the complexity of attempting to merge resolver statements from multiple files in a predictable and useful manner.

  This global TCPIP.DATA file, when specified, will become the first TCPIP.DATA file read regardless of the Socket API library being used. Any parameters found in this file will be global settings for this MVS image. If a global TCPIP.DATA file has been specified then all resolver statements will *only* be obtained from this file. Any of the resolver statements specified in files lower in the search order will be ignored.

  Resolver statements are those required by the resolver to process queries. Resolver TCPIP.DATA statements are:

  – DomainOrigin/Domain
  – NSInterAddr/NameServer
  – NSPortAddr
  – ResolveVia
  – ResolverTimeOut

- ResolverUDPRetries
- Search
- SortList

Other statements not specified in the global TCPIP.DATA file can still be located in one of the TCPIP.DATA files in the search order for each socket API type. For example, if TCPIPJOBNAME is not specified in the global TCPIP.DATA file, the resolver library will locate the next available file in the search order (the search will depend on the socket API being used) and attempt to find the TCPIPJOBNAME there. Note that once a file is found beyond the global TCPIP.DATA file the searching stops. For example, if the TCPIP.DATA file was found by way of the SYSTCPD DD card and no TCPIPJOBNAME was specified in this file, then the normal defaults for TCPIPJOBNAME are applied (for example, TCPIP if the native MVS API search order is used, or a null character if the z/OS UNIX API search order is used). In effect, you can concatenate up to two TCPIP.DATA files with this approach. Note that the search order for the local hosts table (HOSTS.xxxxINFO, ETC.IPNODES, /etc/hosts, or /etc/ipnodes) remains the same. Depending on the application environment, either the native MVS or z/OS UNIX search order will be in effect.

The ability to specify a global TCPIP.DATA file has several advantages. The administrator can decide on which options are global for the installation and which can be specified on an application basis. For example, it is anticipated that most administrators will prefer to control the resolver statements in TCPIP.DATA at a global level. However, it is quite unlikely that they will want a global setting of the TRACE RESOLVER option. This option would typically not be specified on a global TCPIP.DATA file, rather it would get picked up from the first file found in the search order after the global TCPIP.DATA file. This would allow application programmers to continue to turn on the option. Another advantage of this approach is that the administrator may not be aware of all the private TCPIP.DATA files that may be in use on their systems. This approach allows them to implement global options gradually versus an all or nothing approach.

Also, note that this approach lends itself to a multistack (CINET) environment. The administrator can still set up a global TCPIP.DATA file with the global options for this MVS image and omit specifying the TCPIPJOBNAME keyword. The TCPIPJOBNAME keyword would then be located using the appropriate search order. However, using the global TCPIP.DATA file with CINET requires that the resolver TCPIP.DATA statements are able to be used by all stacks. For example, the IP addresses specified by the NameServer statement must be accessible from all stacks. If they are not, then the GLOBALTCPIPDATA file should not be used and you should continue with multiple TCPIP.DATA data sets.

- Support for user specified default TCPIP.DATA file. The DEFAULTTCPIPDATA setup statement identifies the file.

  The user can specify the file to be used as the final location in the search order instead of TCPIP.TCPIP.DATA. This can be used as the replacement for the TCP/IP V3R2 EZAPPRFX sample installation job.

- Specification of the local host file search order for IPv4 and IPv6 name queries. The COMMONSEARCH setup statement identifies that a common local host file search order is to be used for both IPv4 and IPv6 name queries in the native MVS and z/OS UNIX environments. The NOCOMMONSEARCH setup statement identifies that a different local host file search order is to be used for IPv4 and IPv6 name queries in the MVS and UNIX environments.

- Specification of a local host file that contains hard-coded IP addresses and host names that can be used globally. The GLOBALIPNODES setup statement identifies this file.
- Support for a user-specified default local host file. The DEFAULTIPNODES setup statement identifies this file.

Follow the steps in "Setting up a resolver address space" to take advantage of the resolver capabilities described above.

If the setup information is not provided, the resolver uses the applicable native MVS or z/OS UNIX search order without any GLOBALTCPIPDATA, DEFAULTTCPIPDATA, GLOBALIPNODES, DEFAULTIPNODES, or COMMONSEARCH information.

Application programs using the gethostbyaddr and gethostbyname resolver calls from the following IBM APIs result in using the z/OS Communications Server resolver.

- z/OS Language Environment® C/C++ API
- z/OS UNIX Assembler Callable Services
- z/OS Communications Server C/C++ API
- z/OS Communications Server Callable and Macro API
- z/OS Communications Server REXX API
- z/OS Communications Server PASCAL API

Application programs using the getaddrinfo, getnameinfo, and freeaddrinfo resolver calls from the following IBM APIs result in using the z/OS Communications Server resolver.

- z/OS Language Environment C/C++ API
- z/OS UNIX Assembler Callable Services
- z/OS Communications Server Callable and Macro API
- z/OS Communications Server REXX API

Application programs using the sethostent, gethostent, and endhostent resolver calls from the following IBM APIs result in using the z/OS Communications Server resolver.

- z/OS Language Environment C/C++ API
- z/OS Communications Server C/C++ API

The z/OS Communications Server SMTP server, BIND 9 DNS and DNS V9 utilities (dig, nslookup and nsupdate) provide their own unique resolver services. When their resolver initializes it will use GLOBALTCPIPDATA and DEFAULTTCPIPDATA information.

Note that the SMTP resolver only uses the first value of the SEARCH TCPIP.DATA statement when resolving host names.

Setting up a resolver address space

There are two ways in which to start the resolver address space:

- z/OS UNIX initialization will attempt to start the resolver unless explicitly instructed not to. Using z/OS UNIX is the recommended method since it will ensure that the resolver is available before any applications can make a resolution request.

A BPXPRMxx statement, RESOLVER_PROC, is used to specify the procedure name, if any, to be used to start the resolver address space. If the RESOLVER_PROC statement is not in the BPXPRMxx parmlib member or is specified with a procedure name of DEFAULT, z/OS UNIX will start a resolver address space with the assigned name of RESOLVER. The resolver will use the applicable search order for finding TCPIP.DATA statements but without a GLOBALTCPIPDATA specification. If the address space cannot be started, z/OS UNIX initialization continues.

When z/OS UNIX starts the resolver, it is started so that the resolver does not require JES (that is, SUB=MSTR is used). For SUB=MSTR considerations, refer to *z/OS MVS JCL Reference*.

If the RESOLVER_PROC statement has been used to specify a start procedure name, then:

- To find the procedure, it must reside in a data set that is specified by the MSTJCLxx PARMLIB member's IEFPDSI DD card specification. For MSTJCL considerations, refer to *z/OS MVS Initialization and Tuning Reference*.
- The procedure must not contain any DD cards that specify SYSOUT=*.

Since z/OS UNIX does not receive any error indication when it tries to start the address space, it will issue an informational message containing the name of the procedure it has started. The message will be:

```
BPXF224I THE RESOLVER_PROC, procname, IS BEING STARTED.
```

Note: If the RESOLVER_PROC statement is not present or is specified with a procedure name of DEFAULT, *procname* will be RESOLVER even though no start procedure was used. If you want to use the procedure name RESOLVER, a RESOLVER_PROC(RESOLVER) statement must be added to your BPXPRMxx parmlib member.

If the start procedure is not found or has a JCL error in it, the usual z/OS error messages will be issued.

For more detailed information refer to *z/OS UNIX System Services Planning*.

- An installation can use its automation tools to start the resolver by use of the MVS START operator command. If this approach to starting the resolver is used, care should be taken to ensure that no applications that need resolver services (for example, INETD) are started before the resolver address space is initialized. This may mean removing the starting of INETD from the z/OS UNIX /etc/rc file and starting INETD with automation after the resolver has initialized.

Resolver customization

If an installation wants to make use of any resolver setup statement facilities, the following steps will be required. If the facilities are not required, no customization is required and the search order for TCPIP.DATA will be determined by the API being used.

- Create a resolver start procedure

  The procedure requires a //SETUP DD JCL statement that points to a resolver setup file. The z/OS CS provided sample procedure below can be found as member EZBREPRC(alias RESOPROC) in SEZAINST:

```
//RESOLVER PROC PARMS='CTRACE(CTIRES00)'
//*
//*   IBM Communications Server for OS/390
//*   SMP/E distribution name: EZBREPRC
//*
```

```
//*   5694-A01 (C) Copyright IBM Corp. 2001, 2002
//*   Licensed Materials - Property of IBM
//*
//*   Function: Start Resolver
//*
//EZBREINI EXEC PGM=EZBREINI,REGION=0M,TIME=1440,PARM=&PARMS
//*
//*   When the Resolver is started by UNIX System Services it is
//*     started with SUB=MSTR.
//*   This means that JES services are not available to the Resolver
//*     address space. Therefore, no DD cards with SYSOUT can be used.
//*     See the MVS JCL Reference manual for SUB=MSTR considerations in
//*     section "Running a Started Task Under the Master Subsystem".
//*   This Resolver start procedure will need to reside in a data
//*     set that is specified by the MSTJCLxx PARMLIB member's
//*     IEFPDSI DD card specification. If not, the procedure will
//*     not be found and the Resolver will not start.
//*     See the MVS Initialization and Tuning Reference manual for
//*     MSTJCL considerations in section "Understanding the Master
//*     Scheduler Job Control Language"
//*
//*   SETUP contains Resolver setup parameters.
//*     See the section on "Understanding Resolvers" in the
//*     IP Configuration Guide for more information. A sample of
//*     Resolver setup parameters is included in member RESSETUP
//*     of the SEZAINST data set.
//*
//*SETUP   DD   DSN=TCPIP.TCPPARMS(SETUPRES),DISP=SHR,FREE=CLOSE
//*SETUP   DD   DSN=TCPIP.SETUP.RESOLVER,DISP=SHR,FREE=CLOSE
//*SETUP   DD   PATH='/etc/setup.resolver',PATHOPTS=(ORDONLY)
```

- Create a resolver setup file (MVS data set or HFS file)

  The setup file defines the location of the global TCPIP.DATA file (MVS data set or HFS file) and the default TCPIP.DATA name (MVS data set or HFS file). The following statements are supported:

  – comments (; or #)
  – COMMONSEARCH
  – DEFAULTIPNODES
  – DEFAULTTCPIPDATA
  – GLOBALIPNODES
  – GLOBALTCPIPDATA
  – NOCOMMONSEARCH

  The z/OS CS provided sample setup file below can be found as member EZBRECNF(alias RESSETUP) in SEZAINST:

```
;
;   IBM z/OS Communications Server
;   SMP/E distribution name: EZBRECNF
;
;   5694-A01 (C) Copyright IBM Corp. 2002.
;   Licensed Materials - Property of IBM
;
;   Function: Sample Resolver setup file
;
;
;  The following statement defines the final search location for
;   TCPIP.DATA statements. It will replace TCPIP.TCPIP.DATA
;  It may be an MVS data set or HFS file.
;
DEFAULTTCPIPDATA('TCPIP.TCPIP.DATA')
;
#  The following statement defines the first search location for
```

```
#   TCPIP.DATA statements. It may be an MVS data set or HFS file.
;
;  Update with the correct data set or HFS file name
;
; GLOBALTCPIPDATA('TCPCS.SYS.TCPPARMS(GLOBAL)')
;
; GLOBALTCPIPDATA(/etc/tcpipglobal.data)
;
#  The following statement defines the first search location for
#   IPNODES statements. It may be an MVS data set or HFS file.
;
;  Update with the correct data set or HFS file name
;
; GLOBALIPNODES('TCPCS.SYS.TCPPARMS(IPNODES)')
;
; GLOBALIPNODES('TCPCS.ETC.IPNODES')
;
; GLOBALIPNODES(/etc/ipnodes)
;
#  The following statement defines the final search location for
#   IPNODES statements. It may be an MVS data set or HFS file.
;
;  Update with the correct data set or HFS file name
;
; DEFAULTIPNODES('TCPCS.SYS.TCPPARMS(IPNODES)')
;
; DEFAULTIPNODES('TCPCS.ETC.IPNODES')
;
; DEFAULTIPNODES(/etc/ipnodes)
;
#  The following statement defines if the common search order
#   should be used or not.
;
NOCOMMONSEARCH
;
; COMMONSEARCH
;
```

If the resolver setup file is an MVS data set it must be either sequential (PS) or partitioned (PO) organization, fixed (F) or fixed block format (FB), a logical record length (LRECL) between 80 and 256, and have any valid blocksize (BLKSIZE) for fixed block. If the setup file may need to be modified, a member of an MVS partitioned data set is recommended.

If the file is an HFS file, it can reside in any directory. The maximum length of line supported is 256 characters. If the line is greater than 256 it will be truncated to 256 and processed.

The user ID assigned to the resolver address space needs read access (through RACF or equivalent) to SYS1.PARMLIB, the resolver setup file, the global TCPIP.DATA file, the default TCPIP.DATA file, the global IPNODES file, and the default IPNODES file. Likewise, any user IDs or jobs using TCPIP facilities will need read access to the global TCPIP.DATA file, the default TCPIP.DATA file, the global IPNODES file, and the default IPNODES file. For example, for the MVS data set RACF UACC=READ and for the HFS file, permission bits of 644 (Owner can read and write, Group can read, Other can read) could be used. For an HFS file, an OMVS segment or the default OMVS segment must be configured for the resolver user ID and any user IDs or jobs using TCPIP facilities.

- Update the z/OS UNIX BPXPRMxx parmlib member

  The resolver start procedure name should be specified as the *procname* in the BPXPRMxx Parmlib member's RESOLVER_PROC(*procname*) statement. If for

some reason the recommended method of using z/OS UNIX to start the resolver is not desired, use the MVS START command to start the resolver address space.

Managing the resolver address space

A BPXPRMxx statement, RESOLVER_PROC, is used to specify the procedure name, if any, to be used to start the resolver address space. If the RESOLVER_PROC statement is not in the BPXPRMxx parmlib member or is specified with a procedure name of DEFAULT, z/OS UNIX will start a resolver address space with the assigned name of RESOLVER. This name is used with the following MVS system commands to manage the resolver address space:

- Start (S)
- Stop (P)

  Stopping and restarting of the resolver should only be used if a new level of the resolver code has been installed.
- Force
- Modify (F)

  The MODIFY command should be used to dynamically change resolver setup statements, update the resolver's usage of TCPIP.DATA statements, or update the resolver's usage of local host and services tables. Dynamic changes are not supported by the resolver provided by the SMTP server, BIND 9 DNS and DNS V9 utilities.

Refer to *z/OS Communications Server: IP System Administrator's Commands* for command details.

The following MVS System commands can be used to control and display the status of the resolver CTRACE facilities:

- Trace CT
- Display Trace

Refer to the *z/OS Communications Server: IP Diagnosis* for CTRACE usage and control information.

Understanding search orders of configuration information

It is important to understand the search order for configuration files used by TCP/IP functions, and when you can override the default search order with environment variables, JCL, or other variables you provide. This knowledge allows you to accommodate your local data set and HFS file naming standards, and it is helpful to know the configuration data set or HFS file in use when diagnosing problems.

It is important to note that the z/OS CS environment consists of the TCP/IP address space, z/OS CS applications, and the TCP/IP MVS applications. The TCP/IP address space functions are also referred to as the *stack*. The z/OS CS applications refer to those applications using the z/OS UNIX socket API. The TCP/IP MVS applications refer to those applications written to the MVS APIs (for example, C, Sockets-Extended, CICS, IMS, and REXX). The TCP/IP stack and both sets of applications have some common (or global) configuration files, but they also use configuration files that are different.

Another important point to note is that when a search order is applied for any configuration file, the search ends with the first file found. Therefore, unexpected results are possible if you place configuration information in a file that never gets

found, either because other files exist earlier in the search order, or because the file is not included in the search order chosen by the application.

Configuration data set naming conventions

When searching for configuration files, you can explicitly tell TCP/IP where most configuration files are by using DD statements in the JCL procedures or by setting environment variables. Otherwise, you can let TCP/IP dynamically determine the location of the configuration files, based on search orders shown in Table 2 on page 21.

For example, in Table 2 on page 21, for the FTP server application, if the installation did not code the //SYSFTPD DD statement, the FTP server would search for *jobname.*FTP.DATA, then file /etc/ftp.data, then data set SYS1.TCPPARMS(FTPDATA), and finally *hlq*.FTP.DATA.

Dynamic data set allocation

TCP/IP makes extensive use of dynamically allocated data sets using the MVS dynamic data set allocation function to search for configuration files. Multiple versions of a configuration data set can exist, each having a different high-level qualifier or middle-level qualifier. The search order for any configuration file will determine which data set is found and used.

High-level qualifier: High-level qualifiers (HLQ) permit you to associate an application's configuration data set with a particular jobname or TSO user ID, or permit you to use a default configuration data set for the application. The possible high-level qualifiers are:

- *userid*

  Userid is the TSO user ID which invoked the application.
- *jobname*

  Jobname is the application's batch JCL jobname or the name of the application's started procedure.
- *hlq*

  TCP/IP is distributed with a default high-level qualifier (HLQ) of *TCPIP*. To override the default HLQ used by dynamic data set allocation, specify the DATASETPREFIX statement in the TCPIP.DATA configuration file. For most configuration files, the DATASETPREFIX value is used as the high-level qualifier of the data set name in the last step in the search order. Note that the DATASETPREFIX value is not used as the high-level qualifier of the data set name used as the last step in the search order for the PROFILE.TCPIP and TCPIP.DATA configuration files.

Middle-level qualifiers: Multiple middle-level qualifiers (MLQ) permit the isolation of certain profile and translation table data sets. Two of the possible middle-level qualifiers are:

- Node name

  Node name is an MLQ used in the search order for finding the configuration file PROFILE.TCPIP. Node name is determined by the parameters specified during VMCF initialization. For further information on initializing VMCF, refer to *z/OS Program Directory*.
- Function name

  The TCP/IP implementation of national language support (NLS) and double-byte character set (DBCS) support requires the use of multiple translation tables. To facilitate the concurrent use of multiple languages and code pages, TCP/IP uses

a middle-level qualifier to designate which server or client uses a particular translation table. STANDARD, the default MLQ, is available for use if a single translation table can be used by multiple servers or clients. The TCP/IP Telnet client and FTP provide a TRANSLATE parameter that permits you to specify your chosen MLQ to replace the function name for that invocation of the command. For example, SRVRFTP is used as an MLQ by the File Transfer Protocol server.

Following are some of the data sets that are only dynamically allocated by TCP/IP in a configuration file search order (you cannot specify them with DD statements in JCL):

```
ETC.PROTO               ETC.RPC
HOSTS.ADDRINFO          HOSTS.SITEINFO
SRVRFTP.TCPCHBIN        SRVRFTP.TCPHGBIN
SRVRFTP.TCPKJBIN        SRVRFTP.TCPSCBIN
SRVRFTP.TCPXLBIN        STANDARD.TCPCHBIN
STANDARD.TCPHGBIN       STANDARD.TCPKJBIN
STANDARD.TCPSCBIN       STANDARD.TCPXLBIN
```

For each of these data sets, the fully qualified name is established by using one of the following values as the data set HLQ:

- User ID or job name
- DATASETPREFIX value

Naming conventions for dynamically allocated data sets: A data set that you allocate explicitly (with a DD statement in JCL) can have any valid MVS data set name or HFS file name. A data set that you create for the purpose of being allocated dynamically by TCP/IP must use the following naming conventions.

Note: In the examples below, xxxx indicates an appropriate high-level qualifier, yyyy indicates an appropriate middle-level qualifier, and zzzz indicates an appropriate low-level qualifier.

- *userid.yyyy.zzzz*

  userid is the user ID of the logged on TSO user.
- *TSOprefix.yyyy.zzzz*

  TSOprefix is the data set prefix established by the TSO PROFILE command. *userid* is the default value of *TSOprefix*.
- *jobname.yyyy.zzzz*

  jobname is the job name specified on the JOB statement for a job stream or the procedure name for a started procedure.
- *hlq.yyyy.zzzz*

  hlq is the TCP/IP HLQ distributed as the system default, which can be overridden by the value in the DATASETPREFIX statement.
- *xxxx.nodename.zzzz*

  nodename is an MLQ that is used to define the data set name for the TCP/IP stack profile data set.
- *xxxx.function_name.zzzz*

  function_name denotes an acronym specifying a particular TCP/IP server (for example SRVRFTP for the FTP server) and is used as an MLQ for the translation table data set for that application.
- *xxxx.private_name.zzzz*

  private_name is a user-specified private qualifier that can be specified as an option on some TCP/IP commands.
- SYS1.TCPPARMS(TCPDATA)

The member of a system data set used to find the *configuration file* TCPIP.DATA.

Table 2 lists the configuration data sets used by the TCP/IP servers and functions. It includes the name of the sample and the usage of the data set.

Table 2. TCP/IP configuration data sets

Data set (search order)	Copied from	Usage
hlq.ETC.IPNODES	SEZAINST(EZBREIPN)	One of the local host files used for IPv6 name query, or IPv4 and IPv6 name query when COMMONSEARCH is specified in the resolver setup file.
ETC.PROTO	usr/lpp/tcpip/samples/protocol	Used to map types of protocol to integer values to determine the availability of the specified protocol. Required by several z/OS CS components. **Note:** The search order depends on the type of application (z/OS UNIX or native MVS).
ETC.RPC	SEZAINST(ETCRPC)	Defines RPC applications to the Portmapper function.
ETC.SERVICES	usr/lpp/tcpip/samples/services	Establishes port numbers for servers using TCP and UDP. Required for z/OS UNIX SNMP, OROUTED, and OMPROUTE (if the RIP protocol is used). **Note:** The search order depends on the type of application (z/OS UNIX or native MVS).
FTP.DATA 1. //SYSFTPD 2. *userid/jobname*.FTP.DATA 3. /etc/ftp.data 4. SYS1.TCPPARMS(FTPDATA) 5. *hlq*.FTP.DATA	SEZAINST(FTCDATA) for the client and (FTPSDATA) for the server	Overrides default FTP client and server parameters for the FTP server. For more information about *hlq*, *jobname*, or *userid*, see Chapter 9, “Transferring files using FTP” on page 383.
HOSTS.LOCAL (or /etc/hosts)	SEZAINST(HOSTS)	Input data set to MAKESITE for generation of HOSTS.ADDRINFO and HOSTS.SITEINFO.
LPD.CONFIG	SEZAINST(LPDDATA)	Configures the Line Printer Daemon for the Remote Print Server.
LU62CFG	SEZAINST(LU62CFG)	Provides configuration parameters for the SNALINK LU6.2 interface.
MASTER.DATA	No sample provided	DNS database input required for authoritative name servers.
MIBS.DATA 1. The name of an HFS file or an MVS file specified by the MIBS_DATA environment variable 2. /etc/mibs.data HFS file	No sample provided	Defines textual names for MIB objects for the osnmp command.
NPSIDATE	SEZAINST(NPSIDATE)	Operates the TCP/IP X.25 NCP Packet Switching Interface.
NPSIGATE	SEZAINST(NPSIGATE)	Supports GATE MCHs for X.25 NCP Packet Switching Interface.

Table 2. TCP/IP configuration data sets (continued)

Data set (search order)	Copied from	Usage
OMPROUTE configuration 1. The name of an HFS file or MVS file specified by the OMPROUTE_FILE environment variable 2. /etc/omproute.conf 3. *hlq*.ETC.OMPROUTE.CONF	SEZAINST(EZAORCFG)	Contains OMPROUTE configuration statements.
OSNMP.CONF 1. /etc/osnmp.conf 2. /etc/snmpv2.conf	/usr/lpp/tcpip/samples/snmpv2.conf	Defines target host security parameters for the osnmp command.
OSNMPD.DATA 1. The name of an HFS file or MVS file specified by the OSNMPD_DATA environment variable 2. /etc/osnmpd.data HFS file 3. The data set specified on the OSNMPD DD statement in the agent procedure 4. *jobname*.OSNMPD.DATA, where *jobname* is the name of the job used to start the SNMP agent 5. SYS1.TCPPARMS(OSNMPD) 6. *hlq*.OSNMPD.DATA, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used **Note:** The first file found in the search order is used.	/usr/lpp/tcpip/samples/osnmpd.data	Used by SNMP for setting values for selected MIB objects.
PAGENT.CONF 1. File or data set specified with -c startup option 2. File or data set specified with PAGENT_CONFIG_FILE environment table 3. /etc/pagent.conf 4. *hlq*.PAGENT.CONF	/usr/lpp/tcpip/samples/pagent.conf	Defines Policy Agent configuration parameters and optionally defines service policies (rules and actions).
PROFILE.TCPIP 1. //PROFILE 2. *job_name.node_name*.TCPIP 3. *hlq.node_name*.TCPIP 4. *job_name*.PROFILE.TCPIP 5. *hlq*.PROFILE.TCPIP	SEZAINST(SAMPPROF)	Provides TCP/IP initialization parameters and specifications for network interfaces and routing.
Resolver Setup File	SEZAINST (RESSETUP)	Provides configuration statements for the resolver.

Table 2. TCP/IP configuration data sets (continued)

Data set (search order)	Copied from	Usage
PW.SRC 1. The name of an HFS file or an MVS file specified by the PW_SRC environment variable 2. /etc/pw.src HFS file 3. The data set specified on SYSPWSRC DD statement in the agent procedure 4. *jobname*.PW.SRC, where *jobname* is the name of the job used to start the SNMP agent 5. SYS1.TCPPARMS(PWSRC) 6. *hlq*.PW.SRC, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used **Note:** The first file found in the search order is used.	No sample provided	Defines a list of community names used when accessing objects on a destination SNMP agent.
RSVPD.CONF 1. File or data set specified with -c startup option 2. File or data set specified with PAGENT_CONFIG_FILE environment table 3. /etc/rsvpd.conf 4. hlq.RSVPD.CONF	/usr/lpp/tcpip/samples/rsvpd.conf	Defines RSVP Agent configuration parameters.
SNMPD.BOOTS 1. The name of an HFS file or an MVS file specified by the SNMPD_BOOTS environment variable. 2. /etc/snmpd.boots **Note:** The first file found in the search order is used.	No sample provided	Defines the SNMP agent security and notification destinations. **Note:** If the SNMPD.BOOTS file is not provided, the SNMP agent creates the file. If multiple SNMPv3 agents are running on the same MVS image, use the environment variable to specify different SNMPD.BOOTS files for the different agents. For security reasons, ensure unique engine IDs are used for different SNMP agents.
SNMPD.CONF 1. The name of an HFS file or an MVS file specified by the SNMPD_CONF environment variable. 2. /etc/snmpd.conf **Note:** The first file found in the search order is used.	/usr/lpp/tcpip/samples/snmpd.conf	Defines the SNMP agent security and notification destinations. **Note:** If the SNMPD.CONF file is found, the PW.SRC file and the SNMPTRAP.DEST files are not used.

Table 2. TCP/IP configuration data sets (continued)

Data set (search order)	Copied from	Usage
SNMPTRAP.DEST 1. The name of an HFS file or an MVS file specified by the SNMPTRAP_DEST environment variable 2. /etc/snmptrap.dest HFS file 3. The data set specified on SNMPTRAP DD statement in the agent procedure 4. *jobname*.SNMPTRAP.DEST, where *jobname* is the name of the job used to start the SNMP agent 5. SYS1.TCPPARMS(SNMPTRAP) 6. *hlq*.SNMPTRAP.DEST, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used **Note:** The first file found in the search order is used.	No sample provided	Defines a list of managers to which the SNMP agent sends traps.
SMTPCONF	SEZAINST(SMTPCONF)	Provides configuration parameters for the Simple Mail Transfer Protocol.
SMTPNOTE	SEZAINST(SMTPNOTE)	Defines note parameters for Simple Mail Transfer Protocol.
TCPIP.DATA	SEZAINST(TCPDATA)	Provides parameters for TCP/IP client programs. **Note:** The search order depends on the type of application (z/OS UNIX or native MVS).
TNDBCSCN	SEZAINST(TNDBCSCN)	Provides configuration parameters for Telnet 3270 Transform support.
TRAPFWD.CONF 1. An HFS file or an MVS data set specified by the TRAPFWD_CONF environment variable 2. /etc/trapfwd.conf **Note:** The first file found in the search order is used.	No sample provided	Defines addresses to which the Trap Forwarder Daemon forwards traps. **Note:** If the environment variable is set and if the file specified by the environment variable is not found, the Trap Forwarder daemon terminates.
VTAMLST	SEZAINST(VTAMLST)	Defines VTAM® applications and their characteristics. Entries required for Telnet, SNALINK LU0, SNALINK LU6.2, and X.25 NPSI Server.
X25CONF	SEZAINST(X25CONF)	Provides configuration parameters for the X.25 NCP Packet Switching Interface.
X25VSVC	SEZAINST(X25VSVC)	Provides switched virtual circuit configuration for the X.25 NCP Packet Switching Interface.

Configuration files for the TCP/IP stack

Two configuration files are used by the TCP/IP stack, PROFILE.TCPIP and TCPIP.DATA. PROFILE.TCPIP is used only for the configuration of the TCP/IP stack. TCPIP.DATA is used during configuration of both the TCP/IP stack and applications; the search order used to find TCPIP.DATA is the same for both the TCP/IP stack and applications.

PROFILE.TCPIP search order

During initialization of the TCP/IP stack, system operation and configuration parameters for the TCP/IP stack are read from the configuration file PROFILE.TCPIP. As shown in Table 2 on page 21, the search order used by the TCP/IP stack to find PROFILE.TCPIP involves both explicit and dynamic data set allocation as follows:

- //PROFILE DD DSN=*aaa.bbb.ccc(anyname)*
- *jobname.nodename*.TCPIP
- *hlq.nodename*.TCPIP
- *jobname*.PROFILE.TCPIP
- TCPIP.PROFILE.TCPIP

Note: Explicitly specifying the PROFILE DD statement in the TCPIPROC JCL is the recommended way to specify PROFILE.TCPIP. If this DD statement is present, the data set it defines is explicitly allocated by MVS and no dynamic allocation is done. If this statement is not present, the search order continues to use dynamic allocation for the PROFILE.TCPIP.

Examples

The following examples show the search order used by TCP/IP to find the configuration file PROFILE.TCPIP. These examples use the sample TCP/IP started procedure, TCPIPROC, installed in the *hlq*.SEZAINST data set.

Example when DD cards are in your TCP/IP startup procedure: In this example, the PROFILE DD cards are specified as follows:

```
//TCPIP  PROC PARMS='CTRACE(CTIEZB00)'
//*
//* z/OS Communications Server
//* SMP/E Distribution Name: EZAEB01G
//*
//*     5694-A01 (C) Copr. IBM Corp. 1991,2001.
//*     All rights reserved.
//*     US Government Users Restricted Rights -
//*     Use, duplication or disclosure restricted
//*     by GSA ADP Schedule Contract with IBM Corp.
//*     See IBM Copyright Instructions
//*
//TCPIP    EXEC PGM=EZBTCPIP,
//             PARM='&PARMS',
//             REGION=0K,TIME=1440
//*
:
:
//PROFILE DD DISP=SHR,DSN=MVSA.PROD.PARMS(PROFILE)
:
:
```

Because the PROFILE DD is the first step in the search order, TCP/IP uses the data set MVSA.PROD.PARMS(PROFILE) as the PROFILE.TCPIP configuration file.

Example when no DD cards are in your TCP/IP startup procedure: In this example, the PROFILE DD statement is not specified:

```
//TCPIP  PROC PARMS='CTRACE(CTIEZB00)'
//*
//* z/OS Communications Server
//* SMP/E Distribution Name: EZAEB01G
//*
//*     5694-A01 (C) Copr. IBM Corp. 1991,2001.
//*     All rights reserved.
//*     US Government Users Restricted Rights -
//*     Use, duplication or disclosure restricted
//*     by GSA ADP Schedule Contract with IBM Corp.
//*     See IBM Copyright Instructions
//*
//TCPIP    EXEC PGM=EZBTCPIP,
//             PARM='&PARMS',
//             REGION=0K,TIME=1440
//*
:
```

For the configuration file PROFILE.TCPIP, the search order used is as follows:

1. PROFILE DD

   No PROFILE DD exists...search continues.
2. *jobname.nodename.*TCPIP

   If *jobname.nodename.*TCPIP is found, the search stops here.
3. *hlq.nodename.*TCPIP

   If *hlq.nodename.*TCPIP is found, the search stops here.
4. *jobname*.PROFILE.TCPIP

   If *jobname*.PROFILE.TCPIP is found, the search stops here.
5. TCPIP.PROFILE.TCPIP

   TCPIP.PROFILE.TCPIP is searched last if necessary.

TCPIP.DATA search order

TCPIP.DATA is used by the stack address space as follows:

- The TCP/IP stack's configuration component uses TCPIP.DATA during TCP/IP stack initialization to determine the stack's HOSTNAME. To get its value, the z/OS UNIX environment search order is used.
- The TCP/IP stack's TN3270 Telnet server component uses TCPIP.DATA statements to resolve a client's IP address to a name. To obtain the resolver-related statements for address resolution, the native MVS environment search order is used.

For details on the z/OS UNIX environment and native MVS environment search orders and the usage of z/OS UNIX environment variables, see "Resolver configuration files" on page 27.

Configuration files for TCP/IP applications

This section describes the resolver configuration files that can be used by TCP/IP applications and the search orders for those files. In addition to resolver files, an application can also have its own configuration files that are specific to that application. For more information about application-specific configuration files, see the descriptions of the individual applications in Part 2, "Server applications" on page 267.

Resolver configuration files

Understanding the resolver search orders used in native MVS and z/OS UNIX environments is key to setting up your system properly.

As described in “Understanding resolvers” on page 12, the resolver can use available name servers, local definitions, or a combination of both, to process API resolver requests. Figure 1 shows how local definitions can be specified and searched for when needed.

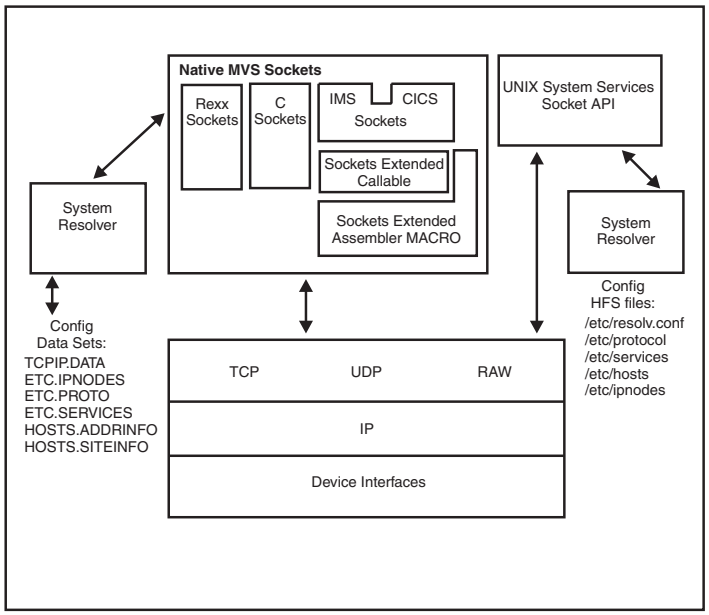


Figure 1. Resolver related configuration files in z/OS UNIX and native MVS environments

Table 3 shows the complete set of local definition possibilities available to the resolver. The actual search order of the candidate files varies depending on the type of API used and the resolver's setup. The search orders are explained in more detail in “Search orders used in the z/OS UNIX environment” on page 28 and “Search orders used in the native MVS environment” on page 33.

Table 3. Local definitions available to resolver

File type description	APIs affected	Candidate files
Base resolver configuration files	All APIs	1. GLOBALTCPIPDATA 2. RESOLVER_CONFIG environment variable 3. /etc/resolv.conf 4. SYSTCPD DD-name 5. *userid*.TCPIP.DATA 6. *jobname*.TCPIP.DATA 7. SYS1.TCPPARMS(TCPDATA) 8. DEFAULTTCPIPDATA 9. TCPIP.TCPIP.DATA

Table 3. Local definitions available to resolver (continued)

File type description	APIs affected	Candidate files
Translate tables	All APIs	1. X_XLATE environment variable 2. *userid*.STANDARD.TCPXLBIN 3. *jobname*.STANDARD.TCPXLBIN 4. *hlq*.STANDARD.TCPXLBIN 5. Resolver-provided translate table, member STANDARD in SEZATCPX
Local host tables	endhostent endnetent getaddrinfo gethostbyaddr gethostbyname gethostent GetHostNumber GetHostResol GetHostString getnameinfo getnetbyaddr getnetbyname getnetent IsLocalHost Resolve sethostent setnetent	1. X_SITE environment variable 2. X_ADDR environment variable 3. /etc/hosts 4. *userid*.HOSTS.xxxxINFO 5. *jobname*.HOSTS.xxxxINFO 6. *hlq*.HOSTS.xxxxINFO 7. GLOBALIPNODES 8. RESOLVER_IPNODES environment variable 9. *userid*.ETC.IPNODES 10. *jobname*.ETC.IPNODES 11. *hlq*.ETC.IPNODES 12. DEFAULTIPNODES 13. /etc/ipnodes
Protocol information	endprotoent getprotobyname getprotobynumber getprotoent setprotoent	1. /etc/protocol 2. *userid*.ETC.PROTO 3. *jobname*.ETC.PROTO 4. *hlq*.ETC.PROTO
Services information	endservent getaddrinfo getnameinfo getservbyname getservbyport getservent setservent	1. /etc/services 2. SERVICES DD-name 3. *userid*.ETC.SERVICES 4. *jobname*.ETC.SERVICES 5. *hlq*.ETC.SERVICES
Host alias table	getaddrinfo gethostbyname	HOSTALIASES environment variable

Search orders used in the z/OS UNIX environment

This section describes setting environment variables for configuration files, and the search orders used in the z/OS UNIX environment for the different file types shown in Table 3 on page 27. The z/OS UNIX socket functions utilize various types of TCP/IP data sets and HFS files. They include:

- Base resolver configuration files
- Translate tables
- Local host tables
- Protocol information
- Services information
- Host alias table

The particular file or table chosen can be either an MVS data set or an HFS file, depending on the resolver configuration settings and the presence of given files on the system.

Note: A program's first resolver service request initializes the resolver definitions that will be used for all resolver requests. For long running programs, the definitions can be modified by use of the MODIFY REFRESH operator command. For command usage and syntax, see *z/OS Communications Server: IP System Administrator's Commands*.

Setting environment variables for configuration files:

A z/OS C/C++ environment variable is an identifier used like a variable in a program. In Table 3 on page 27, the following environment variables appear:

HOSTALIASES
The host aliases data set or file.

RESOLVER_CONFIG
The resolver configuration data sets or files.

RESOLVER_IPNODES
The IPNODES data sets or files.

X_SITE and X_ADDR
The HOSTS.SITEINFO and HOSTS.ADDRINFO data sets or files created by the MAKESITE TSO command.

X_XLATE
The ASCII-EBCDIC translate table data set or file created by the CONVXLAT TSO command.

Setting an environment variable so that a z/OS UNIX application is able to retrieve the value depends on whether the z/OS UNIX application is started from the z/OS shell or from JCL.

If the z/OS UNIX application is to be started from the z/OS shell, the export shell command can be used to set the environment variable. For example, to set the value of RESOLVER_CONFIG to the HFS file /etc/tcpa.data, you can code the following export command:

```
export RESOLVER_CONFIG=/etc/tcpa.data
```

If instead of an HFS file, you want to set RESOLVER_CONFIG to the data set MVSA.PROD.PARMS(TCPDATA), you can specify the following export command. Be sure to put the single quotation marks around the data set name. If you do not, your user ID will be added as a prefix to the data set name when the resolver tries to open the file.

```
export RESOLVER_CONFIG="//'MVSA.PROD.PARMS(TCPDATA)'"
```

If the z/OS UNIX application is to be started from JCL instead of from the z/OS shell, the environment variable needs to be passed as a parameter in the JCL of the application. For example, the following shows the RESOLVER_CONFIG variable set to pick up the TCPIP.DATA information from a file in the HFS:

```
//OSNMPD    PROC
//*
//* Procedure for running the SNMP agent
//*
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/'
```

```
//      'ENVAR("RESOLVER_CONFIG=/etc/tcpa.data")/-d 0'))
:
```

The following example shows the RESOLVER_CONFIG variable set to pick up the TCPIP.DATA information from a partitioned data set:

```
//OSNMPD    PROC
//*
//* Procedure for running the SNMP agent
//*
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/'
//  'ENVAR("RESOLVER_CONFIG=//''TCPA.MYFILE(TCPDATA)''")/-d 0'))
:
```

The following example shows an alternate method of accessing environment variables:

```
//OSNMPD    PROC
//*
//* Procedure for running the SNMP agent
//*
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)/'
// 'ENVAR("_CEE_ENVFILE=DD:STDENV")/-d 0'))
//STDENV DD DSN=TCPA.MYFILE(TCPDATA),DISP=SHR
```

In this case, the environment variables will be read from the file specified on the STDENV DD statement. If this file is an MVS data set, the data set must be allocated with RECFM=V. RECFM=F is not recommended, because RECFM=F enables padding with blanks for the environment variables. See *z/OS C/C++ Programming Guide* for more information on specifying a list of environment variables using the _CEE_ENVFILE environment variable.

Base resolver configuration files: The base resolver configuration file contains TCPIP.DATA statements. In addition to resolver directives, it is referenced to determine, among other things, the data set prefix (DATASETPREFIX statement's value) to be used when trying to access some of the configuration files specified in this section.

The search order used to access the base resolver configuration file is as follows:

1. GLOBALTCPIPDATA

   If defined, the resolver GLOBALTCPIPDATA setup statement value is used. For a description of the GLOBALTCPIPDATA statement, see "Understanding resolvers" on page 12.

   The search continues for an additional configuration file. The search ends with the next file found.
2. The value of the environment variable RESOLVER_CONFIG

   The value of the environment variable is used. This search will fail if the file does not exist or is allocated exclusively elsewhere.
3. /etc/resolv.conf
4. //SYSTCPD DD card

   The data set allocated to the DDname SYSTCPD is used. In the z/OS UNIX environment, a child process does not have access to the SYSTCPD DD. This is because the SYSTCPD allocation is not inherited from the parent process over the fork() or exec function calls.
5. *userid*.TCPIP.DATA

userid is the user ID that is associated with the current security environment (address space or task/thread)

6. SYS1.TCPPARMS(TCPDATA)
7. DEFAULTTCPIPDATA

   If defined, the resolver DEFAULTTCPIPDATA setup statement value is used. For a description of the DEFAULTTCPIPDATA statement, see "Understanding resolvers" on page 12.
8. TCPIP.TCPIP.DATA

Any TCPIP.DATA statements that have not been found will have their default values, if any, assigned.

Translate tables: The translate tables (EBCDIC-to-ASCII and ASCII-to-EBCDIC) are referenced to determine the translate data sets to be used.

The search order used to access this configuration file is as follows. The search order ends at the first file found:

1. The value of the environment variable X_XLATE

   The value of the environment variable is the name of the translate table produced by the CONVXLAT TSO command.
2. *userid*.STANDARD.TCPXLBIN

   userid is the user ID that is associated with the current security environment (address space or task/thread).
3. *hlq*.STANDARD.TCPXLBIN

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.
4. If no table is found, the resolver uses a hardcoded default table that is identical to the STANDARD member in the SEZATCPX data set.

Local host tables: The local host table supplies sitename information for, as one example, resolving hostnames to host or network addresses. The local host table can also supply address information, for example, for resolving addresses to hostname or network names. There are different search orders used for selecting the local host table for these different purposes. The search order to use is based on certain resolver setup statements, the type of API invocation, and possibly the type of host address (IPv4 versus IPv6) being requested or being resolved.

IPv4-unique search order for sitename information: The resolver uses the IPv4-unique search order for sitename information when the resolver setup statement NOCOMMONSEARCH is specified (or left to default), and either the:

- getaddrinfo API is attempting to locate an IPv4 address.
- gethostbyname, sethostent, gethostent, or endhostent API is invoked.

The resolver uses the IPv4-unique search order for sitename information unconditionally for getnetbyname API calls.

The IPv4-unique search order for sitename information is as follows. The search ends at the first file found:

1. The value of the environment variable X_SITE

   The value of the environment variable is the name of the sitename information file created by the TSO MAKESITE command.
2. /etc/hosts

3. *userid*.HOSTS.SITEINFO

   userid is the user ID that is associated with the current security environment (address space or task/thread).
4. *hlq*.HOSTS.SITEINFO

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.

IPv4-unique search order for address information: The resolver uses the IPv4-unique search order for address information when the resolver setup statement NOCOMMONSEARCH is specified (or left to default), and either the getnameinfo API is attempting to resolve an IPv4 address or the gethostbyaddr API is invoked.

The resolver uses the IPv4-unique search order for address information unconditionally for the setnetent, getnetent, endnetent, or getnetbyaddr APIs.

The IPv4-unique search order for address information is as follows. The search ends at the first file found:

1. The value of the environment variable X_ADDR

   The value of the environment variable is the name of the address information file created by the TSO MAKESITE command.
2. /etc/hosts
3. *userid*.HOSTS.ADDRINFO

   userid is the user ID that is associated with the current security environment (address space or task/thread).
4. *hlq*.HOSTS.ADDRINFO

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.

IPv6/common search order: The resolver uses the IPv6/common search order when it determines that any of the following conditions exist:

- The resolver setup statement COMMONSEARCH is specified and the getaddrinfo, gethostbyname, getnameinfo, gethostbyaddr, sethostent, gethostent, or endhostent APIs are invoked.
- The resolver setup statement NOCOMMONSEARCH is specified (or left to default), and the getaddrinfo API is attempting to locate an IPv6 address.
- The resolver setup statement NOCOMMONSEARCH is specified (or left to default), and the getnameinfo API is attempting to resolve an IPv6 address.

Note: The IPv6/common search order is never used for the following API socket calls:

- getnetbyname
- getnetbyaddr
- setnetent
- getnetent
- endnetent

The IPv6/common search order is as follows. The search ends at the first file found:

1. GLOBALIPNODES value

If defined, the resolver GLOBALIPNODES setup statement value is used. For a description of the GLOBALIPNODES statement, see “Understanding resolvers” on page 12.

2. The value of the environment variable RESOLVER_IPNODES
3. *userid*.ETC.IPNODES

   userid is the user ID that is associated with the current security environment (address space or task/thread).
4. *hlq*.ETC.IPNODES

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.
5. DEFAULTIPNODES

   If defined, the resolver DEFAULTIPNODES setup statement value is used. For a description of the DEFAULTIPNODES statement, see “Understanding resolvers” on page 12.
6. /etc/ipnodes

Protocol information: The protocol information supplies protocol related information for the socket calls listed in Table 3 on page 27.

The search order used to access this configuration file is as follows. The search ends at the first file found:

1. /etc/protocol
2. *userid*.ETC.PROTO

   userid is the user ID that is associated with the current security environment (address space or task/thread).
3. *hlq*.ETC.PROTO

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); Otherwise, *hlq* is TCPIP by default.

Services information: The services information supplies the service information for the socket calls listed in Table 3 on page 27.

The search order used to access this configuration file is as follows. The search ends at the first file found:

1. /etc/services
2. *userid*.ETC.SERVICES

   userid is the user ID that is associated with the current security environment (address space or task/thread).
3. *hlq*.ETC.SERVICES

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); Otherwise, *hlq* is TCPIP by default.

Host alias table: The host alias table supplies hostname alias information for the socket calls listed in Table 3 on page 27. The search order used to access this configuration file consists only of the value of the environment variable HOSTALIASES.

Search orders used in the native MVS environment

The native MVS environment socket functions utilize various type of TCP/IP data sets, including:

- Base resolver configuration files

- Translate tables
- Local host tables
- Protocol information
- Services information

The particular file or table chosen depends on the resolver configuration settings and the presence of given files on the system.

Note: A program's first resolver service request initializes the resolver definitions that will be used for all resolver requests. For long running programs, the definitions can be modified by use of the MODIFY REFRESH operator command. For command usage and syntax, see *z/OS Communications Server: IP System Administrator's Commands*.

Base resolver configuration files: The base resolver configuration file contains TCPIP.DATA statements. In addition to resolver directives, it is referenced to determine, among other things, the data set prefix (DATASETPREFIX statement's value) to be used when trying to access some of the configuration files specified in this section.

The search order used to access the base resolver configuration file is as follows:

1. GLOBALTCPIPDATA.

   If defined, the resolver GLOBALTCPIPDATA setup statement value is used. For a description of the GLOBALTCPIPDATA statement, see "Understanding resolvers" on page 12.

   The search continues for an additional configuration file. The search ends with the next file found.
2. //SYSTCPD DD card

   The data set allocated to the DDname SYSTCPD is used.
3. *userid/jobname*.TCPIP.DATA

   userid is the user ID that is associated with the current security environment (address space or task/thread).

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
4. SYS1.TCPPARMS(TCPDATA)
5. DEFAULTTCPIPDATA

   If defined, the resolver DEFAULTTCPIPDATA setup statement value is used. For a description of the DEFAULTTCPIPDATA statement, see "Understanding resolvers" on page 12.
6. TCPIP.TCPIP.DATA

Translate tables: The translate tables are referenced to determine the translate data sets to be used.

The search order used to access this configuration file is as follows. The search order ends at the first file found:

1. *userid/jobname*.STANDARD.TCPXLBIN

   userid is the user ID that is associated with the current security environment (address space or task/thread).

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.

2. *hlq*.STANDARD.TCPXLBIN

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.
3. If no table is found, the resolver uses a hardcoded default table that is identical to the STANDARD member in the SEZATCPX data set.

Local host tables: The local host table supplies sitename information for, as one example, resolving hostnames to host or network addresses. The local host table can also supply address information, for example, for resolving addresses to hostname or network names. There are different search orders used for selecting the local host table for these different purposes. The search order to use is based on certain resolver setup statements, the type of API invocation, and possibly the type of host address (IPv4 versus IPv6) being requested or being resolved.

IPv4-unique search order for sitename information: The resolver uses the IPv4-unique search order for sitename information when the resolver setup statement NOCOMMONSEARCH is specified (or left to default), and either the:

- getaddrinfo API is attempting to locate an IPv4 address.
- gethostbyname, GetHostNumber, GetHostResol, IsLocalHost, Resolve, sethostent, gethostent, or endhostent API is invoked.

The resolver uses the IPv4-unique search order for sitename information unconditionally for getnetbyname API calls.

The IPv4-unique search order for sitename information is as follows. The search ends at the first file found:

1. *userid/jobname*.HOSTS.SITEINFO

   userid is the user ID that is associated with the current security environment (address space or task/thread).

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
2. *hlq*.HOSTS.SITEINFO

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.

IPv4-unique search order for address information: The resolver uses the IPv4-unique search order for address information when the resolver setup statement NOCOMMONSEARCH is specified (or left to default), and either the getnameinfo API is attempting to resolve an IPv4 address or the gethostbyaddr or GetHostString API is invoked.

The resolver uses the IPv4-unique search order for address information unconditionally for the setnetent, getnetent, endnetent, or getnetbyaddr APIs.

The IPv4-unique search order for address information is as follows. The search ends at the first file found:

1. *userid/jobname*.HOSTS.ADDRINFO

   userid is the user ID that is associated with the current security environment (address space or task/thread).

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
2. *hlq*.HOSTS.ADDRINFO

hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.

IPv6/common search order: The resolver uses the IPv6/common search order when it determines that any of the following conditions exist:

- The resolver setup statement COMMONSEARCH is specified, and the getaddrinfo, gethostbyname, getnameinfo, gethostbyaddr, GetHostNumber, GetHostResol, GetHostString, IsLocalHost, Resolve, sethostent, gethostent, or endhostent APIs are invoked.
- The resolver setup statement NOCOMMONSEARCH is specified (or left to default), and the getaddrinfo API is attempting to locate an IPv6 address.
- The resolver setup statement NOCOMMONSEARCH is specified (or left to default), and the getnameinfo or Resolve API is attempting to resolve an IPv6 address.

Note: The IPv6/common search order is never used for the following API socket calls:

- getnetbyname
- getnetbyaddr
- setnetent
- getnetent
- endnetent

The IPv6/common search order is as follows. The search ends at the first file found:

1. GLOBALIPNODES value

   If defined, the resolver GLOBALIPNODES setup statement value is used. For a description of the GLOBALIPNODES statement, see “Understanding resolvers” on page 12.
2. *userid/jobname*.ETC.IPNODES

   userid is the user ID that is associated with the current security environment (address space or task/thread).

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
3. *hlq*.ETC.IPNODES

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, *hlq* is TCPIP by default.
4. DEFAULTIPNODES

   If defined, the resolver DEFAULTIPNODES setup statement value is used. For a description of the DEFAULTIPNODES statement, see “Understanding resolvers” on page 12.
5. /etc/ipnodes

Protocol information: The protocol information supplies protocol related information for the socket calls listed in Table 3 on page 27.

The search order used to access this configuration file is as follows. The search ends at the first file found:

1. *userid/jobname*.ETC.PROTO

   userid is the user ID that is associated with the current security environment (address space or task/thread).

jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.

2. *hlq*.ETC.PROTO

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); Otherwise, *hlq* is TCPIP by default.

Services information: The services information supplies service information for the socket calls listed in Table 3 on page 27.

The search order used to access this configuration file is as follows. The search ends at the first file found:

1. //SERVICES DD card

   The data set allocated to the DDname SERVICES is used.

2. *userid/jobname*.ETC.SERVICES

   userid is the user ID that is associated with the current security environment (address space or task/thread).

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.

3. *hlq*.ETC.SERVICES

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); Otherwise, *hlq* is TCPIP by default.

MVS-related considerations

MVS system symbols

Use of MVS system symbols in the PROFILE.TCPIP and OBEYFILE data sets is automatically supported. This automatic support first tries to use hiperspace memory files to perform the symbol translation, but if an error occurs, then a temporary HFS file will be used. The temporary HFS file is created in either the directory specified by the TMPDIR environment variable or, if the TMPDIR environment variable is not defined, in the /tmp directory.

For MVS system symbols in other configuration files, such as TCPIP.DATA, use the symbol translator utility, EZACFSM1, to translate the symbols before the files are read by TCP/IP. EZACFSM1 reads an input file and writes to an output file, translating any symbols in the process.

Note: The input file and output file can be MVS data sets or HFS files, but do not specify the same file for both the input and output files (this results in a return code of 45 and no translation is attempted).

For more information about the use of MVS system services, refer to *z/OS MVS Initialization and Tuning Guide*.

Following is the symbol translator JCL, found in *hlq*.SEZAINST(CONVSYM), which is used to start EZACFSM1:

```
//________ JOB (accounting,information),programmer.name,
//             MSGLEVEL=(1,1),MSGCLASS=A,CLASS=A
//*
//*   CS for OS/390 IP
//*   SMP/E distribution name: EZACFCSY
//*
//*   5647-A01 (C) Copyright IBM Corp. 1998.
//*   Licensed Materials - Property of IBM
//*
```

```
//*   Function: System Symbols Translator JCL
//*
//*   This JCL kicks off a utility that will read from
//*   an input file that contains MVS System Symbols
//*   and produce an output file which has those symbols
//*   replaced with their substitution text, as defined
//*   in the appropriate IEASYMxx PARMLIB data set; see MVS
//*   Initializaton and Tuning Reference for rules about symbols.
//*
//*   This JCL can be run against any of the TCP/IP configuration
//*   files that contain MVS System Symbols.  An example of how it
//*   could be used is this; a customer could have one base TCPIP.DATA
//*   file containing MVS System Symbols which they edit and maintain.
//*   They would run this utility against this one file the various
//*   MVS systems to produce the TCPIP.DATA file for each different
//*   system.
//*
//STEP1   EXEC  PGM=EZACFSM1,REGION=0K
//SYSIN   DD DSN=TCP.DATA.INPUT,DISP=SHR
//*SYSIN   DD    PATH='/tmp/tcp.data.input'
//*          The input file can be either an MVS file or an HFS file.
//*
//*
//SYSOUT  DD DSN=TCP.DATA.OUTPUT,DISP=SHR
//*SYSOUT  DD    PATH='/tmp/tcp.data.output',PATHOPTS=(OWRONLY,OCREAT),
//*              PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*          The output file can be either an MVS file or an HFS file.
//*
//*          The output file cannot be the same file as the input file-
//*          doing so will result in a return code of 45.
//*
//*          You can mix input and output file types (i.e., the input
//*          can be an MVS file with the output an HFS file, or vice
//*          versa).
//*          Note: Other pathmodes for sysout may be used if needed.
```

The symbol translator utility can be used on any of the TCP/IP configuration files, but because the PROFILE.TCPIP file is automatically translated during TCP/IP initialization, there is no need to run the utility against that file.

Automatic restart manager (ARM)

Automatic restart manager is an MVS component that can automatically restart the TCP/IP stack after an abnormal end (ABEND).

During initialization, TCP/IP automatically registers with the automatic restart manager, using the following options:

```
REQUEST=REGISTER
ELEMENT=EZAsysclonetcpname
```

where:

- *sysclone* is a 1– or 2–character shorthand notation for the name of the MVS system. Refer to *z/OS MVS Initialization and Tuning Guide* for a complete description of the SYSCLONE static system symbol.
- *tcpname* is a 1– to 8–character name of the TCP/IP stack which registers with the automatic restart manager. For example, if the SYSCLONE value is 02 and the TCP/IP stack name is TCPCS, the resulting ELEMENT value is EZA02TCPCS.

```
ELEMTYPE=SYSTCPIP
TERMTYPE=ELEMTERM
```

For more information about automatic restart manager, refer to *z/OS MVS Setting Up a Sysplex*.

Logging of system messages

Syslog daemon (syslogd) is a server process that must be started as one of the first processes in your z/OS UNIX environment. TCP/IP server applications and components use syslogd for logging purposes and can also send trace information to syslogd. Servers on the local system use AF_UNIX sockets to communicate with syslogd; remote servers use AF_INET sockets. z/OS CS components use the **local1**, **daemon**, **mail**, **user**, and **auth** facilities names.

Note: Each application activates and deactivates traces in a slightly different manner. For details, refer to the chapter on the individual application in this document.

The syslog daemon reads and logs system messages to the MVS console, log files, SMF, other machines, or users as specified by the configuration file. If syslogd is not started, log data from some applications will be displayed on the MVS console. For more information on syslogd, refer to Chapter 3, “Customization” on page 101.

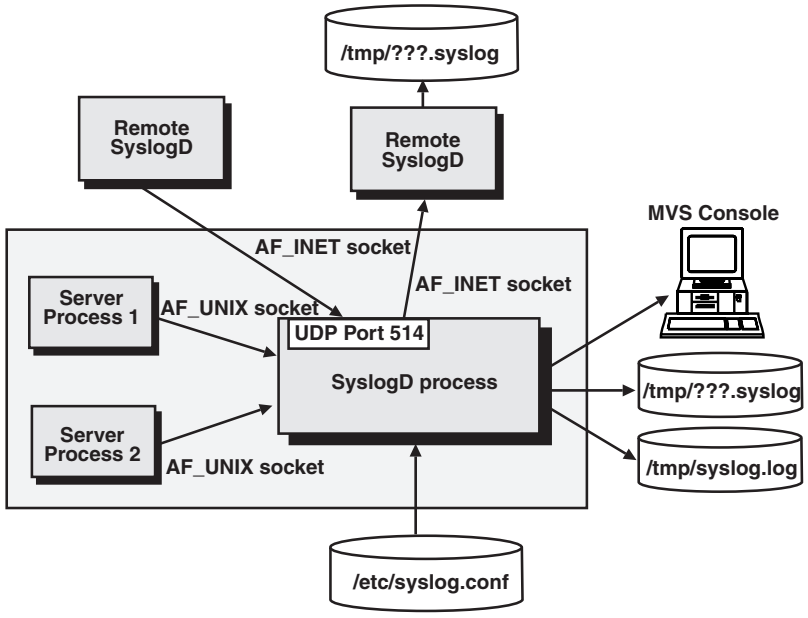


Figure 2. syslogd operation

Note: /tmp/???.syslog is the file specified in the syslogd.conf file.

The syslogd facility uses a common mechanism for segregating messages. Table 4 on page 40 shows the facilities used by z/OS CS functions which write messages to syslogd. The Primary syslog facility column shows the syslog facility used for most messages logged by the application. Some applications use other facilities for certain messages. Table 4 on page 40 also shows any additional facilities.

Table 4. syslogd facilities

Application	syslogd record identifications	Primary syslog facility	Other syslog facility
OTELNETD	telnetd	local1	auth
SENDMAIL	sendmail	mail	None
POPPER	popper	mail	None
ORSHD	rshd	daemon	auth
TCP/IP Configuration	Config	daemon	None
FTP Server	ftpd, ftps	daemon	None
Traffic Regulation Management Daemon (TRMD)	TRMD	daemon	None
ROUTED	routed	daemon	None
NAMED	named	daemon	None
Trap Forwarder Daemon	trapfwd	daemon	None
OREXECD	rexecd	daemon	auth
Policy Agent (PAGENT)	Pagent	daemon	None
Service Level Agreement SNMP Subagent	PASubA	daemon	None
SNMP Agent (OSNMPD)	snmpagent	daemon	None
PWCHANGE Command	pwchange	daemon	None
PWTOKEY Command	pwtokey	daemon	None
syslogd	syslogd	daemon	None
DHCP Server	dhcpsd	user	None
TIMED Daemon	timed	user	None
TFTP Server	tftpd	user	None
OMPROUTE	omproute	user	None
OPORTMAP Server	oportmap	daemon	None

Accounting - SMF records

Installations may use Systems Management Facilities (SMF) records for various purposes such as:

Performance management
Performance management includes the tasks that are related to verifying that defined service levels are met, and if not, identifying possible causes.

Aggregated information about delivered service, structured by organizational units (for which service levels have been defined) is needed to perform these tasks. These reports are typically time series with varying levels of time intervals, ranging from weeks through days to a time interval that matches the SMF interval. Some examples of potential reports related to performance management are:

- TCP connection elapsed time per server port number per time of day (potentially broken down on source IP address, or netmask)
- Number of TCP connections per server port number per time of day (potentially broken down on source IP address, or netmask)
- Number of inbound/outbound bytes transferred in TCP connections per time of day (potentially broken down in various ways: per destination or source port, per destination IP address, netmask, or in total, etc.)
- TCP retransmission activity per time of day (potentially broken down per destination IP address, or netmask)
- Number of outbound TCP connections per time of day (potentially broken down per destination IP address, or netmask)
- Number of inbound/outbound UDP datagrams per time of day (potentially broken down on server port number)
- Number of discards, error packets, and unknown protocol packets inbound and outbound per time of day (potentially broken down per interface)

Capacity planning

Capacity planning includes tasks that are related to forecasting capacity in terms of central processing power, memory, channel-based I/O subsystem, network attachments, and network bandwidth. Such planning tasks are based on analyzing trends for use of capacity during a preceding period (typically one to two years), and applying forecasting metrics, along with knowledge about planned launches of new applications or use of existing applications, to this trend in order to predict capacity needs during the next one to two year period. Some examples of potential reports related to capacity planning are:

- Total number of TCP connections per reserved server port number per day including analysis of average and variations around average during daily peak periods
- Total number of UDP inbound/outbound UDP datagrams per reserved server port number per day including average and variations around average during daily peak periods
- Number of bytes and/or packets transferred inbound and outbound per interface (LINK) per time of day (potentially broken down into unicasts, broadcasts, and multicasts)
- Size of queue length per interface per time of day

Auditing

Auditing involves tasks that are related to identifying and proving that individual events have taken place. Some examples of potential reports related to auditing are:

- Detailed information about specific TCP connections or UDP sockets, IP addresses, server/client identification, duration, number of bytes, etc.
- Details about activity that involves a specific client or server
- Details about a given application session based on server-specific SMF recording, such as individual TN3270 sessions or FTP sessions

Accounting

Accounting involves tasks that are related to calculating how much each individual user or organizational unit should be charged for use of the shared central IS resources. Input to such calculations vary, but is often based on CPU cycle use, data quantities, bandwidth usage, and memory use. For TCP/IP additional metrics may be defined, such as type of service used (FTP, LPD, Web server, etc), and TCP connection-related information

(number of connections, duration, byte transfer counts, etc). Some examples of potential reports related to accounting are:

- Aggregated number of connections to a given server from a given source in terms of a specific client IP address, or netmask
- Accumulated connect time to a given server from a given source in terms of a specific client IP address, or netmask.
- Number of bytes transferred to or from a given source in terms of a specific client IP address, or netmask.
- Application-level accounting information specific to each individual server, for example:
  - For FTP: number of transfer operations and bytes retrieved or stored per user ID
  - For TN3270: number of sessions and session-type (TN3270/TN3270E/LINEMODE)

In general, SMF records are created for deferred processing and analysis. SMF recording is generally not used for real-time monitoring purposes. In a TCP/IP environment, real-time monitoring is implemented using the SNMP protocol and is based on internal variables that are maintained by SNMP subagents, but on z/OS a lot of the information that is written in SMF records is useful from a real-time monitoring perspective, too.

As can be seen from the above, all disciplines require detailed data as input. Depending on the discipline, certain levels of aggregation is performed on the raw detailed data in order to perform the tasks of that discipline. The objective of the TCP/IP product is to define and generate the lowest level of detail that is needed by all disciplines. How to aggregate and the actual aggregation is performed by other products, such as Performance Reporter for z/OS (PR), MVS Information Control System (MICS), or SAS-based tools or, in many cases, customer-written programs.

TCP/IP– produced SMF records should not be viewed isolated. Other components in MVS produce SMF records for the same purposes as those produced by TCP/IP. An installation is likely to combine information from a series of subsystems in performing detailed performance, or capacity planning. SMF records with information about use of CPU resources and memory resources per address space is, for example, produced by other components in MVS, and TCP/IP produced SMF records should not duplicate that information.

The events that trigger SMF records to be written and the information included in the SMF records must accommodate the intended purposes. There can be multiple purposes for given SMF records.

SMF records can be cut at multiple levels in the TCP/IP protocol stack, and the type of information that can be included depends on where the SMF record is created:

- At the IP and interface layer we know about ICMP activity, IP packet fragmentation and reassembly activity, IP checksum errors, IGMP activity, and ARP activity. At this layer, it is difficult to relate the information to specific users (remote clients, local socket address spaces, and so on), so from an accounting point of view, this information is not very interesting. From a performance and capacity planning point of view, this information is of interest because it allows the installation to aggregate network-layer activity to physical interfaces, which is an important aspect of both performance and capacity management.
- At the transport protocol layer, we know about IP addresses, port numbers, and host names. For TCP related work, we know about connections and information

that is related to TCP connections, such as byte counts, connection times, reliability metrics, and performance metrics. For UDP related work, each UDP datagram is a separate entity, and the only way we can aggregate information for UDP is on a UDP socket-level, where we could cut SMF records every time a UDP socket is closed.

- At the application layer, we know more details about what goes on, but every application is different and it requires separate SMF record definitions and ability to write the SMF records to implement application-layer SMF recording. We currently do it for the stack Telnet server and the FTP server, but not for any other servers.

SMF accounting issues (Record type 118)

Many installations rely on the MVS component SMF for job accounting and for performance analysis. TCP/IP can create SMF type 118 records for certain events. If you are running multiple stacks, SMF does not always allow you to distinguish among them. Consider the following issues:

- There is no stack identity in SMF type 118 records. SMF records that are written by the system address space or by standard servers may be identified as belonging to one stack or another, based on address space naming conventions.
- SMF records written by client address spaces cannot be identified as belonging to a single stack based on the address space naming conventions used in standard servers.
- The only technique currently available to distinguish among records written by various client address spaces is to assign unique SMF type 118 record subtype intervals to each stack:
  - FTP server: One or nine subtypes in FTP.DATA
  - Telnet server: Two subtypes on TELNETPARMS
  - API: Two subtypes on SMFPARMS
  - FTP, Telnet client: One subtype on SMFPARMS

If you choose to assign subtypes, there will be an obvious impact on your local accounting programs. SMF type 118 subtype changes and additions must be coordinated with persons responsible for managing the use of SMF.

SMF type 118 records do not support IPv6 addresses. Thus, if you choose to exploit IPv6 in your environment, migrate your SMF processing to use the SMF type 119 records, which do support IPv6 addresses.

An external mapping (EZASMF76 macro) is available for customers to parse the SMF type 118 records that TCP/IP generates. EZASMF76 produces assembler level DSECTs for the Telnet (server and client), FTP (server and client), and API SMF records.

Note: If the BPX.SMF facility is defined and SMF records are to be written by syslogd, the user ID with which syslogd runs must be permitted to BPX.SMF.

To create the Telnet SMF Record layout, code:

```
EZASMF76 TELNET=YES
```

To create the FTP SMF Record layout, code:

```
EZASMF76 FTP=YES
```

To create the API SMF Record layout, code:

```
EZASMF76 API=YES
```

SMF accounting issues (Record type 119)

SMF type 119 records contain unique stack identification sections designed to eliminate the confusion of the type 118 records. They provide uniformity of date and time (UTC), common record format (self-defining section and TCP/IP identification section) and room to expand to IPv6 addresses and expanded field sizes (64 bit versus 32 bit) for some counters. The kinds of SMF type 119 records available are:

- TCP connection initiation and termination
- UDP socket close
- TCP/IP, interface and server port statistics
- TCP/IP stack start/stop
- FTP server transfer completion
- FTP server logon failure
- FTP client transfer completion
- TN3270 server session initiation and termination
- Telnet client connection initiation and termination.

The SMF type 119 records utilize a common structure. Each record is organized as follows:

- SMF header
- Self-defining section containing pointers to:
  - TCP/IP identification section (identifies system, stack etc)
  - Sections containing the data for the record

An external mapping (EZASMF77 macro) is available for customers to parse the SMF type 119 records that TCP/IP generates.

For more detailed information refer to *z/OS Communications Server: IP Configuration Reference*.

For more information about SMF, refer to *z/OS MVS System Management Facilities (SMF)*.

Security considerations

z/OS CS relies on a System Authorization Facility (SAF) to protect several resources:

- Started tasks require access to a STARTED resource. This is documented in the server information in the *z/OS Communications Server: IP Configuration Reference*. Also, refer to SEZAINST(EZARACF) for SAF authorizations required for the TCP/IP stack and servers started tasks.
- Restricting access to a network, subnetwork or particular IP address in the network is provided by resources in the SERVAUTH class. Using NETACCESS statements, z/OS CS can map networks, subnetworks and IP addresses to SAF resource names. Users that are not permitted access to a particular SAF resource are not allowed to communicate with the corresponding network, subnetwork, or IP address. Refer to the NETACCESS statement in *z/OS Communications Server: IP Configuration Reference* or “Setting up SAF Server Access Authorization (SERVAUTH) (optional)” on page 143 for more information.

  Restricting users’ ability to run applications that access specific TCP and UDP ports is also provided by resources in the SERVAUTH class. z/OS CS provides a one-to-one mapping between port numbers and SAF resource names. Refer to the PORTACCESS statement in the *z/OS Communications Server: IP*

Configuration Reference or "Setting up SAF Server Access Authorization (SERVAUTH) (optional)" on page 143 for more information.

Also, similar to PORTACCESS, z/OS CS ensures a user attempting to connect to a TN3270 secure port is allowed access to the port. This support is used in conjunction with TN3270 SSL client authentication support. Refer to the CLIENTAUTH statement in the *z/OS Communications Server: IP Configuration Reference* or "Setting up SAF Server Access Authorization (SERVAUTH) (optional)" on page 143 for more information.

Restricting access to the TCPIP stack is also controlled under z/OS CS by defining a resource in the SERVAUTH class. Refer to "Setting up SAF Server Access Authorization (SERVAUTH) (optional)" on page 143 for more information.

- Restricting access to operator commands is provided through the OPERCMDS resource. z/OS CS verifies that users have access to specific OPERCMDS resources before executing the operator command. Refer to the operator commands information in the *z/OS Communications Server: IP System Administrator's Commands* or "Setting up SAF Server Access Authorization (SERVAUTH) (optional)" on page 143 for more information about limiting access to z/OS CS commands.
- Restricting access to the TSO and UNIX shell Netstat command is provided by SERVAUTH resources. z/OS CS verifies that users have access to specific SERVAUTH resources before executing the Netstat command. Refer to the Netstat command information in the *z/OS Communications Server: IP System Administrator's Commands* for more information about limiting access to Netstat command. The security product resource names in the SERVAUTH class do not apply to DISPLAY TCPIP,,NETSTAT command. If you wish to restrict access to DISPLAY TCPIP,,NETSTAT command, you can do so using standard operator command restriction facility, OPERCMDS class profiles. Refer to *z/OS MVS Planning: Operations* for more information.

UNIX System Services security considerations

This section describes some of the changes that have a product-wide effect. For descriptions of changes that affect specific servers or components, see the sections of this document that describe each server and component.

Requirement for an OMVS segment

Many TCP/IP Services components in z/OS CS now exploit z/OS UNIX services in both the native MVS environment and in the z/OS UNIX environment. For example, all TCP/IP socket APIs and TCP/IP applications (whether they are provided by z/OS CS, OS/390, other IBM and non-IBM products, or written by users) now make use of z/OS UNIX services.

Use of z/OS UNIX services requires a z/OS UNIX security context, referred to as an *OMVS segment*, for the user ID associated with any unit of work requesting these services. In other words, most user IDs requiring access to TCP/IP functions now require an OMVS segment to be defined in Resource Access Control Facility (RACF).

Note: The tasks, examples, and references in this section assume that you are using the z/OS CS Security Server (RACF). If you are using a security product from another vendor, read the documentation for that product for instructions on task performance.

To satisfy the requirement for an OMVS segment in RACF, do one of the following:

- Identify all the users in your environment that use TCP/IP services and then define OMVS RACF segments for the associated user IDs.
- Use the default OMVS segment support provided by RACF and z/OS UNIX for users and groups.

The default OMVS segments reside in the USER profile and GROUP profile. The names of these profiles are identified by the installation, using the BPX.DEFAULT.USER facility class profile. The application data field in the class profile contains the user ID, or the user ID/group ID, that is used to access the default OMVS segments for users and groups, respectively.

Notes:

1. An HFS must be defined for the OMVS segment, and the home directory must exist.
2. If you use a trusted or privileged started task in ICHRIN03 or the STARTED class (especially a generic entry), be careful in assigning a default UID and GID with facility class BPX.DEFAULT.USER. Whenever trusted or privileged is specified, all default tasks have superuser authority.

To set up default OMVS segments, follow the steps in the Table 5.

Table 5. Setting up default of OMVS segment

Task	Details
Define a Group ID (GID) to the system to be used as an anchor for a default OMVS group segment.	Use the following command: `ADDGROUP DEFGRP OMVS(GID(777777))` Make the GID unique so that it is easily identifiable. The GID can be either very high or very low. The other fields related to the GID are not likely to be used for anything.

Define a user ID (UID) to be used as an anchor for the default OMVS user segment.	Use the following commands:
	`ADDUSER DEFUSR DFLTGRP(DEFGRP) NAME('DEFAULT USER')`
	`OMVS(UID(999999) HOME('/') PROGRAM('/bin/sh'))`
	
	Note: To avoid giving superuser authority, do not use 0 as the UID.
	
	When defining a UID, consider the following:
	• **UID** should be unique so that it is easily identifiable. The number can be very high or very low.
	• **HOME** — Use one of the following options when defining the home directory for the default user:
	– Define the HOME directory as the root (/). The users do not have write access. They do not need to update their home directory.
	– Define the HOME directory in the /tmp directory.
	– Define a directory as you would for any other user. This directory is then used concurrently by many users that do not have an OMVS segment. (*Not recommended*)
	• **PROGRAM** defines the default shell in this field.
	
	The other fields related to this UID are not likely to be used for anything.

Table 5. Setting up default of OMVS segment (continued)

Task	Details

Set up a default for the USER OMVS segment or set up a default UID and GID.	• To set up a default for the USER OMVS segment only, create a facility class profile named BPX.DEFAULT.USER, and then specify the default UID in the application data field. Use the following commands:
	`RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('DEFUSR')`
	`SETROPTS RACLIST(FACILITY) REFRESH`
	Note: You cannot set up a default GROUP OMVS segment alone.
	• To set up a default UID and GID, create a facility class profile named BPX.DEFAULT.USER, and then specify the default UID and GID in the application data field. Use the following commands:
	`RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('DEFUSR/DEFGRP')`
	`SETROPTS RACLIST(FACILITY) REFRESH`
	Be aware that the facility class must be activated. In addition, the USER profile of the default UID and the GROUP profile of the default GID must exist, and must contain OMVS segments with a UID and GID, respectively.
	Note: RACF does not check to ensure that the application data points to a valid UID or UID and GID, or that the USER and GROUP profiles contain OMVS segments with the required UID and GID.

The following process shows how the BPX.DEFAULT.USER facility class profile works:

1. A user requests a UNIX service, which is serviced by the kernel.
2. The kernel calls the security product to extract the UID, GID, HOME, and PROGRAM information.
3. The security product attempts to extract the OMVS segment associated with the user. If the user is not defined, the security product attempts to extract and use the OMVS segment for the default user that was listed in the BPX.DEFAULT.USER profile.

A similar process is followed to obtain a GID when the user default group does not have an OMVS segment.

Authorization of TCP/IP started task user ID

The TCP/IP address space operates as a transport provider for the INET physical file system. For this to occur, the TCP/IP system address space must connect to z/OS UNIX and become a z/OS UNIX process. Therefore, the started task UID that is assigned to the TCP/IP system address space must have a valid OMVS segment.

As a transport provider, the TCP/IP address space requires superuser privileges in z/OS UNIX. Define the TCP/IP system address space started task UID as UID=0, or define the TCP/IP system address space as a trusted environment in the RACF started class profile for the TCP/IP system address space. Use the following command to assign an OMVS segment to the TCP/IP started task user ID specified as UID=0:

```
ALU tcpip_userid OMVS(UID(0) HOME(/) PGM(/bin/sh))
```

Other user IDs requiring z/OS UNIX superuser authority

When a started procedure is used to start the following servers, daemons, and agents, the user must be a superuser [UID(0)] or permitted to BPX.SUPERUSER class profile.

- File transfer protocol (FTP) daemon
- Domain name system (DNS) server
- OROUTED server
- SNMP agent (OSNMPD)
- Network Print Facility (NPF) queue manager

The following daemons are managed by the inetd server, and the user specified in file /etc/inetd.conf must be defined to RACF with UID(0). For details on inetd, refer to *z/OS UNIX System Services Planning*. For details on individual daemons, refer to the *z/OS Communications Server: IP Configuration Reference*.

- z/OS UNIX remote execution daemon (REXECD)
- z/OS UNIX remote shell daemon (RSHD)
- z/OS UNIX Telnet daemon

BPX.DAEMON facility class

Certain z/OS CS TCP/IP Services servers need to change the security environment of the process in which they currently execute. For example, the FTPD daemon creates a new z/OS UNIX process for every FTP client connecting to it. After the new process is created, the daemon changes the security environment of the process so that it is associated with the security context of the logged-in user. The RACF facility class resource BPX.DAEMON is used for this purpose.

Table 6. BPX.DAEMON

Task	Details
Decide if you want to activate the BPX.DAEMON level of security by reviewing the section about BPX.DAEMON authority in *z/OS UNIX System Services Planning* to determine whether this level of security is appropriate for your installation.	This is not required. It is recommended, however, because it provides additional security in the z/OS UNIX environment. The following TCP/IP Services servers and daemons in z/OS CS change the security environment of their processes: • z/OS UNIX TELNETD • z/OS UNIX RSHD • z/OS UNIX REXECD • FTPD
Plan the time at which you define BPX.DAEMON carefully.	As soon as you define the BPX.DAEMON resource, MVS will not let programs change the security environment unless the programs are retrieved from a program-controlled library and unless the UID under which the program executes has access to BPX.DAEMON.
If you decide *not* to define the BPX.DAEMON facility class, assign UID(0) for the UIDs associated with these servers and daemons.	This is sufficient for processing. It is described in “Other user IDs requiring z/OS UNIX superuser authority”.

Table 6. BPX.DAEMON (continued)

Task	Details
If you decide **to** define the BPX.DAEMON facility class, grant READ access to this profile for the UIDs associated with the listed daemons. Also, enable BPX.DAEMON security by defining the BPX.DAEMON facility class profile in RACF	To define the BPX.DAEMON facility class profile in RACF, use the following command: `RDEFINE FACILITY BPX.DAEMON UACC(NONE)` **Note:** You must specify the name BPX.DAEMON in this command. Substitutions for the name are not allowed.

If all the required conditions are not met, your server programs will fail as soon as you define BPX.DAEMON. If the server programs fail, delete BPX.DAEMON, and the setup reverts to its previous state. Check all your definitions, and make the required corrections before trying to define BPX.DAEMON again.

If this is the first facility class profile that your installation is using, activate the facility class using the following commands:

```
SETROPTS CLASSACT(FACILITY) GENERIC(FACILITY) AUDIT(FACILITY)
SETROPTS RACLIST(FACILITY)
```

If you start server programs using MVS start commands or from shell scripts that execute after startup of z/OS UNIX, you must allow the UIDs access to the BPX.DAEMON facility class resource. The following example shows the UID (ftpd_user_ID) with which you can start the FTPD daemon:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(ftpd_user_ID) ACCESS(READ)
```

Authorization to change the user security environment is granted only if both of the following two conditions are true:

- The server program is executing under a UID that has READ permission to the BPX.DAEMON facility class profile and a UID=0.
- All programs running in the address space have been retrieved from a controlled library. Program control is discussed in the following section.

Program control

In a z/OS UNIX environment, there are additional security concerns related to the HFS and the loading of programs that are considered trusted. Program control facilities in RACF and z/OS UNIX provide a mechanism for ensuring that the z/OS UNIX program loading process has the same security features that APF authorization provides in the native MVS environment.

It is recommended that you enable program control in your installation. If you define the BPX.DAEMON facility class, then you *must* enable program control for certain z/OS CS load libraries. Review the section on program control in *z/OS UNIX System Services Planning* to decide whether program control is appropriate for your installation.

To enable program control, follow the tasks in the following table.

Table 7. Program control

Task	Details
Activate program control.	Use the following command: `SETROPTS WHEN(PROGRAM)`

Table 7. Program control (continued)

Task	Details

Set the universal access for public library data sets (those in LINKLST*xx*) to READ. This allows access to the controlled programs and any other program in those libraries. (MVS opens the LNKLST*xx* libraries during IPL and makes these programs public. However, users cannot make changes.)	Use the following commands to create RACF data set profiles:
	`ADDSD 'cee.version.SCEERUN' UACC(READ)`
	`ADDSD 'SYS1.LINKLIB' UACC(READ)`
	`ADDSD 'TCPIP.SEZALOAD' UACC(READ)`
	`ADDSD 'TCPIP.SEZATCP' UACC(READ)`

Ensure all load modules that are loaded by the BPX.DAEMON servers into an address space come from controlled libraries.	If the MVS contents supervisor loads a module from a noncontrolled library, the address space becomes *dirty* and loses its authorization. To prevent this from happening, define all the libraries from which load modules can be loaded as program controlled. At a minimum, this should include the C run-time library, the TCP/IP Services SEZALOAD and SEZATCP libraries, SYS1.LINKLIB, and any load libraries containing FTP security exits.
	
	Use the following commands:
	`RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'/'volser'/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM('cee.version.SCEERUN'/'volser'/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM('TCPIP.SEZALOAD'/'volser'/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM('TCPIP.SEZATCP'/'volser'/NOPADCHK) UACC(READ)`
	`RALTER PROGRAM * ADDMEM('db2.DSNLOAD'/'volser'/NOPADCHK UACC(READ)`
	`RALTER PROGRAM * ADDMEM('db2.DSNEXIT'/'volser'/NPPADCHK UACC(READ)`
	`RALTER PROGRAM * ADDMEM('ftp.userexits'/'volser'/NOPADCHK UACC(READ)`
	
	Note: If you define the load libraries as controlled, do not specify a universal access of NONE for the PROGRAM resources. If you do so for your SYS1.LINKLIB programs, you cannot IPL your MVS system. Be aware also that in , the volser specification is optional.

Activate RACF changes.	Use the following command: `SETROPTS WHEN(PROGRAM) REFRESH`

Defining TCP/IP as a UNIX System Services physical file system (PFS)

As described in *z/OS Communications Server: IP Migration*, the TCP/IP Services stack in z/OS CS must be defined as a z/OS CS UNIX System Services PFS before it can be started. This involves updating the BPXPRM*xx* parmlib member. The following sample definition in BPXPRM*xx* defines TCP/IP as a z/OS CS UNIX System Services PFS, where the network layer is IP Version 4 (IPv4) and communication at the sockets layer is through the AF_INET family:

```
FILESYSTYPE TYPE(INET)  ENTRYPOINT(EZBPFINI)

NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(60000)
        TYPE(INET)
```

The sample definition above shows how to define a single TCP/IP stack as IPv4 only. To define a single TCPIP stack as both IPv4 and IPv6, add an additional NETWORK statement in the BPXPRM*xx* member. The following sample definition in BPXPRM*xx* defines TCP/IP as a z/OS CS UNIX System Services PFS, where the network layer is IP Version 6 (IPv6) and communication at the sockets layer is through the AF_INET6 family:

```
NETWORK DOMAINNAME(AF_INET6)
        DOMAINNUMBER(19)
        MAXSOCKETS(60000)
        TYPE(INET)
```

The BPXPRM*xx* member contains additional z/OS CS UNIX System Services parameters that are crucial to the proper operation of TCP/IP. Carefully examine and specify these parameters:

- MAXPROCSYS — Specifies the maximum number of z/OS UNIX processes that the system allows.
- MAXPROCUSER — Specifies the maximum number of processes associated with a single z/OS CS UNIX System Services user ID.
- MAXUIDS — Specifies the maximum number of z/OS UNIX user IDs that can operate concurrently.
- MAXFILEPROC — Specifies the maximum number of z/OS CS UNIX System Services file descriptors a z/OS CS UNIX System Services process can allocate. This includes access to both HFS files and z/OS CS UNIX System Services socket descriptors. In z/OS CS, most TCP/IP applications access z/OS CS UNIX System Services sockets, either directly or indirectly, using the TCP/IP socket APIs. You should set the MAXFILEPROC value high enough to accommodate the largest number of sockets a single TCP/IP application (or z/OS CS UNIX System Services process) can allocate.

  Be aware that the tn3270 Telnet server is exempt from the limit specified in this parameter. The tn3270 Telnet server can obtain the maximum number of socket connections for a single z/OS CS UNIX System Services process.
- MAXPTYS — Specifies the maximum number of pseudo-terminals for the system.
- MAXTHREADTASKS — Specifies the maximum number of MVS tasks that a single process can have concurrently active.
- MAXTHREADS — Specifies the maximum number of threads that a single process can have concurrently active.
- MAXQUEUEDSIGS — The sum of MAXQUEUEDSIGS and MAXFILEPROC multiplied by 2 is the system wide maximum for the total number of asynchronous z/OS UNIX socket calls that can be outstanding. When specifying this number, consider the following:
  - For every TCP/IP connection that the TN3270 Telnet server has, there is an asynchronous z/OS UNIX socket call outstanding. This is true for both TN3270 and TN3270E clients.
  - Any TCP/IP application, IBM or vendor supplied, that uses either the z/OS UNIX Assembler Callable Services asyncio call or the TCPIP provided Sockets Extended asynchronous API could have one or more outstanding asynchronous socket calls.

The MAXSOCKETS() parameter specifies the total number of z/OS CS UNIX System Services sockets that can be active at any one time. You must ensure that this specification is large enough to accommodate your installation's workload. For example, each connection to your tn3270 Telnet server or FTP server requires one z/OS CS UNIX System Services socket. Once the maximum number of sockets is allocated, then no more Telnet sessions, FTP sessions, or other applications that require z/OS CS UNIX System Services sockets can be started.

Note: If multiple NETWORK statements are defined, MAXSOCKETS can be specified for each NETWORK statement and will be enforced separately.

References

For details on the BPXPRM*xx* member, please refer to the following guides:

- *z/OS UNIX System Services Planning*
- *z/OS MVS Initialization and Tuning Reference*
- *z/OS UNIX System Services File System Interface Reference*

Performance considerations

Follow the guidelines found in the *z/OS MVS Initialization and Tuning Reference*. If your installation is running Workload Manager, follow the guidelines found in *z/OS MVS Planning: Workload Management*.

It is necessary that VTAM, TCPIP and some associated server applications are able to obtain cycles in order to maintain their network presences. In general, we recommend VTAM and TCPIP have a higher dispatching priority than the applications that use their services. Server applications such as OMPROUTE, OROUTED and FTPD should be set at or just below TCPIP's value. If running WLM, these tasks should be assigned to the SYSSTC service class. Additionally, making these tasks non-swappable will assure that they will be available during periods of high CPU usage.

Fast path support

For applications that have extremely strict communications path-length requirements, an optional extension has been provided to further reduce overhead resulting from the z/OS UNIX-to-TCP/IP stack communications. This extension is only available to applications using the UNIX System Services (USS) Callable Services Socket API or the C/C++ socket API supported by the Language Environment (LE). It is not available to applications using the native MVS socket APIs (such as C/C++, EZASMI macro, EZASOKET, REXX, or CICS socket APIs) provided by the Communications Server. Exploitation of this extension is entirely optional.

This feature can be activated for an entire USS process using the z/OS UNIX _BPXK_INET_FASTPATH environmental variable. The value of this variable determines whether a socket application is marked *fast path*. A C/C++ LE application can set the variable by invoking the setenv() service, or you can export the variable to the z/OS UNIX shell environment before the socket application is invoked. An application using the USS Callable Services APIs can set this variable using the BPX1ENV service.

Note: z/OS UNIX environmental variables have a process-wide scope only—that is, they usually affect a single MVS address space only. It is possible, however, to have multiple UNIX processes within a single address space. In this scenario, the setting of this environmental variable might vary for each process within the address space. It is not a problem if some of your applications exploit fast path services, while others do not. When a socket application is marked as fast path, the communications overhead is reduced on the following socket syscalls:

- - send()
- - recv()
- - sendto()
- - recvfrom()

- - sendmsg()
- - recvmsg()

Although applications are more efficient when using the environmental variable, they are not XPG compliant, and POSIX signals are not supported. Applications can be interrupted only with the SIGKILL terminating signal, and they cannot be debugged using the interactive z/OS UNIX dbx debugger. You can, however, develop and test an application using the dbx debugger without setting the environmental variable, and then execute the application in production with the environmental variable set. Also, note that applications using the USS asynchronous socket interface (BPX1AIO) to invoke synchronous socket operations (that is, setting the AioSync bit in the AIOCB) cannot use the BPX1AIO service to cancel outstanding synchronous calls on sockets that are marked as fast path. Doing so will cause the cancel operation to hang.

For environments that do not use common INET, the value of this variable should be set to the name specified on the FILESYSTYPE TYPE() parameter in the BPXPRMxx parmlib member.

For common INET environments, the value used to set the environmental variable depends on whether the application is using the TCP or UDP protocols. In a common INET environment, the variable should be set as follows:

- For UDP applications, it should be set to the name of the TCP/IP stack as specified on the SUBFILESYSTYPE NAME() parameter in the BPXPRMxx parmlib member. The socket application is explicitly associated with the TCP/IP stack named in the environmental variable (that is, the TCP/IP stack name). This means that the socket application can communicate with partners that are accessible only through the specific TCP/IP stack interfaces. For UDP, the environmental variable effectively overrides the support provided by common INET. You should take this contingency into account before activating fast path for a UDP-based application.

  Note that if the UDP application already establishes affinity to a specific TCP/IP stack using other means, such as setting the _BPXK_SETIBMOPT_TRANSPORT environment variable, using setibmopt(), BPX1PCT, and so on, the setting of the fast path variable is ignored. As a result, UDP applications that require fast path support and affinity to a specific TCP/IP stack must do so using the _BPXK_INET_FASTPATH environmental variable.
- For TCP applications, the variable can be set to an asterisk (*), indicating that any TCP/IP stack in the common INET configuration can be used. This allows all TCP/IP stacks that support the fast path model to obtain the fast path performance benefits automatically. TCP servers are not bound to a specific TCP/IP stack, even if they specify a specific TCP/IP stack name on the environmental variable; instead, they can listen for inbound connections across all TCP/IP stacks. When a connection arrives from the TCP/IP stack named in the environmental variable [at the time of the accept()], it is automatically marked as fast path. Connections that arrive from TCP/IP stacks that are not named by the current environmental variable value are not marked as fast path.

  Note, however, that certain TCP/IP API functions, such as the resolver services [that is, gethostbyname(), gethostbyaddr(), getaddrinfo(), and getnameinfo()] and the network interface identification services [that is, if_nameindex(), if_nametoindex(), and if_indextoname()] use UDP sockets internally to perform their processing. Consequently, if a specific TCP/IP stack name is specified on the environmental variable, these hidden UDP sockets will only be associated with the named TCP/IP stack, which might have undesirable effects. For example, any resolver API queries resulting in communications with a domain

name server will occur only over the specified TCP/IP stack. As a result, it is strongly recommended that TCP applications set the environmental variable to the special asterisk (*) value. If the application requires affinity to a specific TCP/IP stack, it should do so using any of the facilities that are provided by USS, such as setibmopt(), BPX1PCT, and so on. For more details on establishing affinity to a specific TCP/IP stack, refer to *z/OS UNIX System Services Planning*.

Applications can also enable fast path processing for a single socket by issuing the Iocc#FastPath IOCTL for the socket, using the w_ioctl() or the BPX1IOC APIs. Note that this IOCTL is only effective if it is issued against a socket that is already associated with a specific TCP/IP stack. Sockets are considered associated with a specific TCP/IP stack if they meet any of the following conditions:

- The application has explicit process affinity to a specific TCP/IP stack [that is, by setting the _BPXK_SETIBMOPT_TRANSPORT environmental variable, using setibmopt(), BPX1PCT, and so on].
- TCP/IP stack affinity has been explicitly established for this socket (that is, using the SIOCSETRTTD IOCTL).
- A bind() has already been issued for the socket using a specific IP address (that is, not INADDR_ANY).
- A TCP (that is, streams) socket that is connected. This includes TCP sockets that are returned as a result of accept() or sockets that a connect() was issued for.

The Iocc#FastPath constant is defined in the BPXYIOCC. Note that this IOCTL requires a 4-byte argument as input. This argument should be set to a nonzero value to activate fast path, or a zero value to disable fast path on the specified socket.

Considerations for multiple instances of TCP/IP

The z/OS Communications Server TCP/IP stack is a multiple-processor capable stack, which means that it can concurrently exploit all available processors on a system. Starting multiple stacks will not yield a significant increase in throughput.

In addition, running multiple z/OS Communications Server TCP/IP stacks requires additional system resources, such as storage, CPU cycles, and DASD. It also adds a significant level of complexity to the system administration tasks for TCP/IP.

For these reasons, it is suggested that in most cases you use the INET configuration, which supports a single TCP/IP stack. However, there are some special situations where running multiple stacks can provide a benefit. For example, you might want to run two separate stacks for intranet and Internet traffic, or AnyNet® Sockets over SNA in conjunction with one or more TCPIP stacks.

Common INET physical file system (CINET PFS)

If you wish to run multiple z/OS Communications Server TCP/IP stacks concurrently, you must use the Common INET (CINET) configuration. In this configuration, up to a maximum of eight TCP/IP stacks can be active at any time.

When the CINET configuration is used, the CINET PFS is inserted between the LFS and the TCP/IP PFS for each stack. The CINET PFS maintains an internal copy of each TCP/IP stack's IP configuration, so that it can preroute a socket call to the correct TCP/IP stack. This allows most socket programs to run with multiple stack

support with no change to the application. In addition, CINET supports IPv6, and is capable of supporting underlying TCP/IP stacks in IPv4/IPv6 dual mode or in IPv4-only mode.

You can specify your choice of INET (single stack) or CINET (multiple stack) support on the NETWORK, DOMAINNAME, FILESYSTYPE, and SUBFILESYSTYPE statements of SYS1.PARMLIB(BPXPRMxx). For more information about the BPXPRMxx statements, refer to "Specifying BPXPRMxx values for a CINET configuration" on page 64 and *z/OS UNIX System Services Planning*.

Port management overview

When there is a single transport provider, and the relationship of server to transport provider is 1:1, port management is relatively simple. Using the PORT statement, the port number can be reserved for the server in the PROFILE.TCPIP for that single transport provider.

Port management becomes more complex in a CINET environment where there are multiple transport providers (multiple instances of TCP/IP) and a potential for multiple combinations of the same server (for example, z/OS UNIX and TN3270/TN3270E Telnet).

In a multiple transport provider environment, the following questions need to be answered for each server in an installation:

- Is the server generic so that it can communicate with multiple TCP/IPs or does the server have an affinity for one instance of the transport providers and can only communicate with one TCP/IP?
- How can ports be reserved across multiple transport providers? When is the port reservation determined by MVS rather than by the job name, procedure name, or user ID?
- How can you synchronize between BPXPARMS and PORTRANGE for ephemeral port reservation?
- How can TCP/IP distinguish between two different instances of Telnet (z/OS UNIX Telnet and TN3270/TN3270E Telnet)?

Generic server versus server with affinity for a specific transport provider

The following sections describe the differences between generic servers and servers with affinities for specific transport providers.

Generic server: A generic server, a server without an affinity for a specific transport provider, provides service to any client on the network. (See Figure 3 on page 56.) FTP is an example of a generic server. The transport provider is merely a connection linking client and server. The service File Transfer is not related to the internal functioning of the transport provider, and the server can communicate concurrently over any number of transport providers.

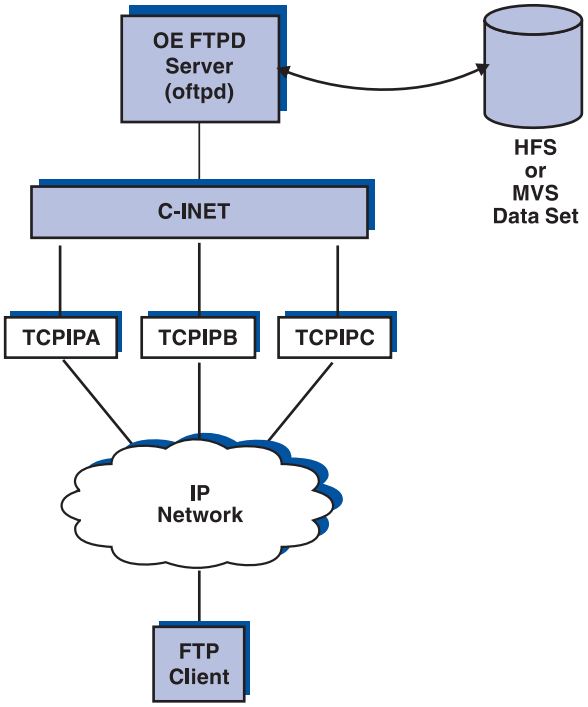


Figure 3. Generic server

Server with an affinity for a specific transport provider: When the service is related to the internal functioning of the transport provider (for example, Telnet, OMPROUTE, OSNMPD, and the command, onetstat), there must be an explicit binding of the server application to the chosen transport provider. (See Figure 4 on page 57.) There must also be a way to specify the single transport to be chosen.

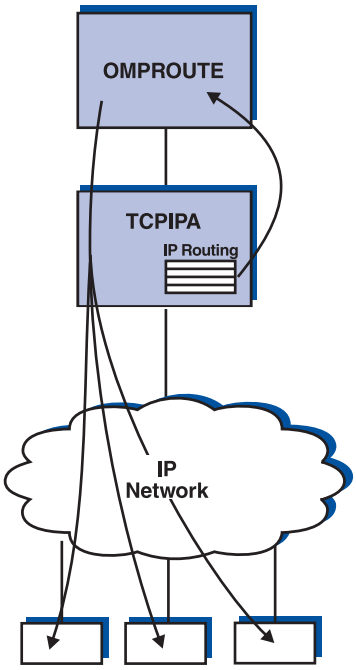


Figure 4. Server with affinity for a specific transport provider

With the exception of applications that use the socket API provided by TCP/IP, other IBM-supplied applications that use the z/OS UNIX socket API and that must bind to a specific transport provider use the z/OS UNIX socket call `setibmopt()` (refer to *z/OS C/C++ Run-Time Library Reference*) to specify which TCP they have chosen. A C function `__iptcpn()`, described in the *z/OS C/C++ Run-Time Library Reference*, enables the application to search the TCPIP.DATA file to find the name of the specific TCP/IP. (See Figure 5.) An application that uses the z/OS LE runtime can also establish stack affinity by setting the environment variable _BPXK_SETIBMOPT_TRANSPORT.

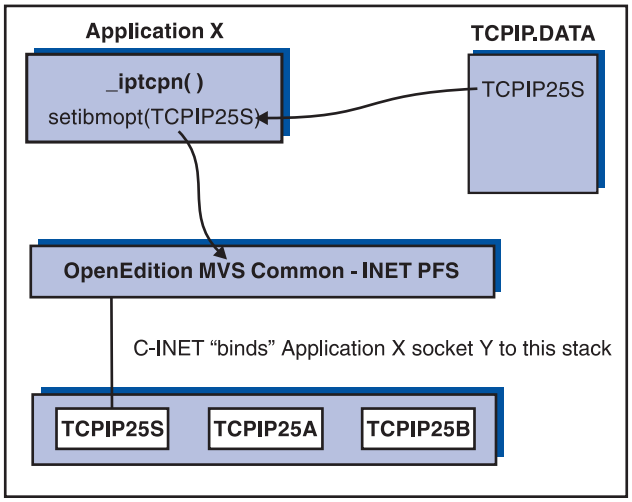


Figure 5. Example of binding an application to a specific transport provider

Generic servers in a CINET environment

In z/OS CS, you can configure multiple TCP/IP stacks in a single MVS image using the CINET feature. In a CINET configuration, an application using the z/OS UNIX socket interface can get transparent access to all the TCP/IP protocol stacks configured under CINET. For example, when an application coded to z/OS UNIX sockets performs a SOCKET/BIND/LISTEN in a CINET environment, the request is propagated by CINET to all the TCP/IP stacks. This application can then service client requests that arrive into any of the configured TCP/IP stacks without having any awareness of this fact. This type of application is often referred to as a *generic server* or *daemon*.

The following servers or daemons shipped by z/OS CS are generic:

- FTPD
- z/OS UNIX RSHD
- z/OS UNIX REXECD
- z/OS UNIX TELNETD
- z/OS UNIX SENDMAIL
- z/OS UNIX POPPER
- TFTPD
- TIMED
- z/OS UNIX Portmap

z/OS UNIX RSHD, REXECD and TELNETD are usually started by the INETD daemon, which is shipped as part of the z/OS UNIX. Because INETD is also a generic daemon, any server processes started by INETD inherently become generic servers as well.

If a server started by INETD (a generic server) requires affinity to a specific stack, this affinity can be accomplished by use of the _BPXK_SETIBMOPT_TRANSPORT environment variable. For more information about the _BPXK_SETIBMOPT_TRANSPORT environment variable refer to *z/OS UNIX System Services Planning*.

The _BPXK_SETIBMOPT_TRANSPORT environment variable, when set, has an effect similar to the setibmopt() function call provided by the C/C++ compiler and described in the *z/OS C/C++ Run-Time Library Reference*. This variable can be set in the JCL for a started procedure or batch job that executes a z/OS UNIX C/C++ program to indicate which TCP/IP stack instance the application should bind to. TCP/IP applications that require affinity to a specific TCP/IP stack, like OSNMPD and OROUTED, use the setibmopt() function call directly. The _BPXK_SETIBMOPT_TRANSPORT environment variable basically provides the ability to bind a generic server type of application to a specific stack.

For example, if you had two TCP/IP stacks configured under CINET, one named TCPIP and the other TCPIPOE, and you wanted to start an FTPD server instance that was associated with TCPIPOE, you could modify the FTPD procedure as follows:

```
//FTPD   PROC MODULE='FTPD',PARMS='TRACE'
//FTPD   EXEC PGM=&MODULE,REGION=7M,TIME=NOLIMIT,
//       PARM=('POSIX(ON) ALL31(ON)',
//      'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE")',
//      '/&PARMS')
//CEEDUMP  DD SYSOUT=*
//SYSFTSX DD DISP=SHR,DSN=TCPV34.STANDARD.TCPXLBIN
```

All the parameters specified prior to the slash (/) in the parameter statement are processed by the C/C++ run time library. Parameters to be passed to the FTPD program must appear after the slash (/). Also note how the parameters were split over three lines in this example because they could not fit on a single line.

The following example uses JCL for the started procedure for INETD:

```
//INETD  PROC
//*************************************************************
//INETD  EXEC PGM=BPXBATCH,
//*      PARM='PGM /usr/sbin/inetd -d /etc/inetd.conf'
//       PARM='PGM /usr/sbin/inetd    //''USER1.INETD.CONF'''
//*
//STDERR  DD PATH='/tmp/inetd.debug.stderr',
//  PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//  PATHMODE=SIRWXU
//STDOUT  DD PATH='/tmp/inetd.debug.stdout',
//  PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//  PATHMODE=SIRWXU
//STDENV   DD  DISP=SHR,DSN=USER1.INETD.ENVIRON
```

The STDENV data set would contain the _BPX_SETIBMOPT_TRANSPORT variable as follows:

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE
```

In the previous examples, INETD was also passed its configuration file as a parameter. In our examples, this file is an MVS data set rather than an HFS file; therefore, it requires the additional double slash (//) and quotes that the example shows.

Multiple instances of INETD are not allowed, even if each instance is bound to a different TCP/IP stack. This is an INETD restriction, not a TCP/IP restriction. Therefore, if you decide to make INETD have affinity to a specific stack, then that is the only INETD instance that you will be able to have running in that MVS image.

Notes:

1. The _BPXK_SETIBMOPT_TRANSPORT variable should be specified only for a generic server type of application.

   If specified for a non-generic server and/or non-z/OS UNIX application it will not have any effect.

2. The name specified for _BPXK_SETIBMOPT_TRANSPORT must match the job name associated with the TCP/IP stack.

   If the name specified does not match the job name of any TCP/IP stacks defined for CINET, the application will receive a z/OS UNIX return code of X'3F3' and a return value of X'005A' and may be accompanied by the following message:

   ```
   EDC8011I A name of a PFS was specified that either is
   not configured or is not a Sockets PFS.
   ```

   If the name specified does not match the job name of any currently active TCP/IP stack defined under CINET, the application will receive a z/OS UNIX return code of X'70' and a return value of X'0296' and may be accompanied by the following message:

   ```
   EDC5112I Resource temporarily unavailable.
   ```

3. For more detailed information about requesting transport affinity, refer to *z/OS UNIX System Services Planning*.

Port reservation across multiple transport providers

When there are multiple transport providers, be sure to synchronize the PORT statements in each of the PROFILE.TCPIP files to ensure that the port reservations for each stack match the port definitions for the servers that will be using that stack.

For more information about reserving ports with the PORT statement, see Chapter 3, “Customization” on page 101.

Ephemeral ports: When running with multiple transport providers, just as it is necessary to synchronize PORT reservations for specific applications across all stacks, it is required to synchronize reservations for port numbers that will be dynamically assigned across all stacks. These are the ephemeral ports above 1023, which are assigned by the stack when none is specified on the application `bind()`. To reserve a group of ports in the PROFILE.TCPIP, use PORTRANGE. For more information about PORTRANGE, see Chapter 3, “Customization” on page 101. Specify the same PORTRANGE for every stack. In addition, you need to let the z/OS UNIX CINET know which ports are guaranteed to be available on every stack. The following is an example of reserving ports 4000 to 4999 in the two required files:

- PROFILE.TCPIP
  - PORTRANGE 4000 1000 TCP OMVS ; Reserved for OMVS
  - PORTRANGE 4000 1000 UDP OMVS ; Reserved for OMVS
- BPXPRMxx parmlib member
  - NETWORK DOMAINNAME(AF_INET)
  - INADDRANYPORT(4000)
  - INADDRANYCOUNT(1000)

Note: When IPv6 is configured and there are two NETWORK statements, INADDRANYPORT and INADDRANYCOUNT only need to be specified for the NETWORK statement for AF_INET and not for AF_INET6. If they are specified for AF_INET6, they are ignored and the values from the NETWORK statement for AF_INET are used if provided. Otherwise, the default values are used.

Selecting a stack when running multiple instances of TCP/IP

Socket application programs in a multi-stack (CINET) environment must contend with the following:

- How the socket program selects which TCP/IP stack to use for its socket communication
- How the TCP/IP resolver code executing in the socket application address space decides which TCP/IP resolver configuration data sets to allocate

Note: If a resolver GLOBALTCPIPDATA setup file is used, a local TCPIP.DATA cannot override any explicit statements in the global file and cannot override any resolver statements. Therefore, in a CINET environment, the TCPIPJOBNAME statement should not be specified in the GLOBALTCPIPDATA file. Also, using the GLOBALTCPIPDATA file with CINET requires that the resolver TCPIP.DATA statements are able to be used by all stacks. For example, the IP addresses specified by the NameServer statement must be accessible from all stacks. If they are not, then the GLOBALTCPIPDATA file should not be used and you should continue with multiple TCPIP.DATA data sets. For details, see “Understanding resolvers” on page 12.

To answer these questions, a distinction must be made between standard servers and clients (those that come with the z/OS CS product), and other socket application programs, including those you might have written yourself.

Standard servers and clients

The anchor configuration data set is the TCPIP.DATA data set. This is the base resolver configuration data set with information on host name, domain origin, and so on. It holds the TCPIPJOBNAME statement, which identifies the TCP/IP stack to use, and the DATASETPREFIX statement, which is used by the resolver code and other services when allocating configuration data sets. For more information on these data sets, see “Configuration files for TCP/IP applications” on page 26.

The key to selecting both a specific stack and resolver configuration data sets is to control which TCPIP.DATA data set a standard server or client address space allocates. Applications that use the z/OS UNIX API can use Common INET to determine which stack an application will use. But, it is important to ensure that the search order and the contents of the resolver configuration data set are understood.

Native MVS servers and clients search for TCPIP.DATA in sequences as described in “Search orders used in the native MVS environment” on page 33.

z/OS UNIX servers and clients will search for TCPIP.DATA in sequences as described in “Search orders used in the z/OS UNIX environment” on page 28.

Nonstandard servers and clients

Nonstandard servers and clients (those that do not come with the z/OS CS product) also use TCPIP.DATA to decide which resolver configuration data sets to allocate. Depending on the socket API used, they might or might not use the TCPIPJOBNAME parameter to select a stack.

If you run sockets programs from other products or vendors, you may want to know which sockets API was used to develop the program, and which techniques, if any, the program uses to specify the name of the TCP/IP system address space. As long as application programs that use a TCP/IP socket library do not specify anything specific on calls `setibmopt()`, `Initialize`, or `INITAPI`, the TCPIPJOBNAME from a TCPIP.DATA data set will be used for finding a TCP/IP system address space name.

Table 8 depicts the differences that prevail in stack selection depending on the TCP/IP socket API under which you are running the socket program.

Table 8. How your own socket programs select a stack

C sockets	Callable and Macro	Pascal sockets	REXX sockets
SETIBMOPT or TCPIPJOBNAME from TCPIP.DATA	TCPNAME on INITAPI or TCPIPJOBNAME from TCPIP.DATA	TCPIPJOBNAME from TCPIP.DATA	Service on Initialize or TCPIPJOBNAME from TCPIP.DATA
Callable and Macro programs might have a configuration option to specify the TCP/IP system address space name, or might interrogate the available stacks via the getibmopt() call.			

A Callable or Macro program does not have to call INITAPI. If INITAPI is not called, an implicit INITAPI is performed with the value taken from TCPIPJOBNAME in a

TCPIP.DATA data set. If INITAPI is called with the TCPNAME parameter specified as a space, the TCP/IP system address space name results in the TCPIPJOBNAME keyword value.

In a z/OS UNIX INET (single stack) environment, the socket application program is always associated with the single TCP/IP stack. In the z/OS UNIX Common INET (CINET) environment, your application will be associated with multiple TCP/IP stacks unless the application specifically associates with a particular stack using the z/OS UNIX socket call setibmopt(). For other ways of requesting stack affinity in a CINET environment, refer to *z/OS UNIX System Services Planning*.

TCP/IP TSO clients

TSO client functions can be directed against any of a number of TCP/IP stacks. Obviously, the client function must be able to find the TCPIP.DATA appropriate to the stack of interest at any one time. Some TSO client commands provide a parameter to specify the stack to be used. For those that do not, the following methods are available for finding the relevant TCPIP.DATA:

- Add a SYSTCPD DD statement to your TSO logon procedure. The issue with this approach is that a separate TSO logon procedure per stack is required, and users have to log off TSO and log on again using another TSO logon procedure in order to switch from one stack to another.
- Use one common TSO logon procedure without a SYSTCPD DD statement. Before a TSO user starts any TCP/IP client programs, the user has to issue a TSO ALLOC command wherein the user allocates a TCPIP.DATA data set to DD name SYSTCPD. To switch from one stack to another, the user simply has to deallocate the current SYSTCPD allocation (for example, TSO FREE command) and allocate another TCPIP.DATA data set.
- Combine the first and second methods. Use one logon procedure to specify a SYSTCPD DD for a default stack. To switch stacks, issue TSO ALLOC to allocate a new SYSTCPD. To switch back, issue TSO ALLOC again with the name that was on the SYSTCPD DD in the logon procedure. The disadvantage to this approach is that the name that was on the SYSTCPD DD is hidden in the logon procedure and needs to be retrieved or remembered.

The last method can be implemented by creating a small REXX program for every TCP/IP stack on your MVS system. For each stack create a REXX program with the name of the stack (for example, T18A or T18B). Whenever TSO users want to use the T18A stack, they run the T18A REXX program. Any TCP/IP functions invoked thereafter will use the T18A stack for socket communication. If users want to switch to the T18B stack, they run the T18B REXX program. See Figure 6 on page 63 for an example.

```
/* REXX  "T18B" */
/*********************************************************************/
/*                                                                   */
/* Switch TSO Address Space to use the T18B Stack.                   */
/* Subsequent NETSTAT command will be directed toward                */
/*  the T18BTCP stack.                                               */
/*                                                                   */
/*********************************************************************/
Say 'Switching to T18BTCP stack'

msgstat = msg()
z = msg("OFF")
"FREE FI(SYSTCPD)"
"ALLOC FI(SYSTCPD) DA('TCPIP.T18B.TCPPARMS(TCPDATA)') SHR"
z = msg(msgstat)

exit(0)
```

Figure 6. REXX program to switch TSO user to another TCP/IP stack

Selecting configuration data sets

The resolver code and other services that execute as part of the socket program address space to service calls such as `gethostbyname()`, `getservbyname()` and `getprotobyname()` allocate one or more resolver configuration files to service these calls. All socket programs, including standard servers and clients and homegrown socket programs, need access to resolver configuration files. For information on how the resolver configuration files are found and used, see "Configuration files for TCP/IP applications" on page 26.

Sharing resolver configuration data sets

The general recommendation is to use separate DATASETPREFIX values for each stack and create separate copies of the required configuration data sets; at the very least, create separate copies of the resolver configuration data sets. For a test and a production stack, however, you would probably use different DATASETPREFIX values. However, if the stacks are functionally identical, you may share the same DATASETPREFIX values and many of the same configuration data sets. You need separate TCPIP.DATA data sets because of the two different TCPIPJOBNAMEs. On the other hand, you may choose to share the resolver configuration data sets between the stacks by using the same DATASETPREFIX value in each TCPIP.DATA data set.

In addition to separate TCPIP.DATA data sets, separate /etc/resolv.conf files might also be necessary. If this is the case, use the environment variable RESOLVER_CONFIG to point to the appropriate resolver information.

Exercise caution if servers use DATASETPREFIX to allocate server-specific configuration data sets. Try to use explicit allocation as far as possible in your server JCL procedures. Most servers allow you to explicitly allocate their configuration data sets using DD statements.

Some servers may use DATASETPREFIX to create new data sets. Servers that do create new data sets allow you to specify an alternate data set prefix for the data sets that are created. NPF creates new sequential data sets with captured print data. NPF has a special keyword in NPF.DATA for this purpose; it is called

NPFPRINTPREFIX. If this keyword is specified, NPF will use that as the high-level qualifier for newly created print data sets instead of taking the DATASETPREFIX value from TCPIP.DATA. Another example of a server that creates new data sets is the SMTP server.

Specifying BPXPRMxx values for a CINET configuration

For a detailed description of parameters in SYS1.PARMLIB(BPXPRMxx), refer to *z/OS UNIX System Services Planning* and *z/OS MVS Initialization and Tuning Guide*.

```
/* AF_INET file system for sockets      */
/* CINET support - BPXTCINT  */

FILESYSTYPE TYPE(CINET)
            ENTRYPOINT(BPXTCINT)  1
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(10000)  2          TYPE(CINET)
        INADDRANYPORT(10000)  3
        INADDRANYCOUNT(2000)
NETWORK DOMAINNAME(AF_INET6) 4
        DOMAINNUMBER(19)
        MAXSOCKETS(10000)  2          TYPE(CINET)
SUBFILESYSTYPE NAME(TCPIP1A)  5
               TYPE(CINET)
               ENTRYPOINT(EZBPFINI)  6
               DEFAULT  7
SUBFILESYSTYPE NAME(TCPIP1B)  5
               TYPE(CINET)
               ENTRYPOINT(EZBPFINI)
```

Figure 7. SYS1.PARMLIB(BPXPRMxx) for CINET

1 CINET and BPXTCINT specify the use of CINET.

2 The MAXSOCKETS operand specifies the maximum number of sockets that can be obtained for the given file system type. It should be large enough for the number of sockets needed for applications using z/OS CS. MAXSOCKETS is enforced independently for AF_INET (IPv4 sockets) and AF_INET6 (IPv6 sockets).

3 INADDRANYPORT and INADDRANYCOUNT specify the first ephemeral port number and the range of ports for z/OS UNIX. These values have to match the PORTRANGE definitions in your PROFILE data sets for both TCP/IP stacks. INADDRANYPORT and INADDRANYCOUNT should not be specified in the NETWORK statement for AF_INET6. If these parameters are specifed in the NETWORK statement for AF_INET6, they are ignored. The INADDRANYPORT and INADDRANYCOUNT values for AF_INET6 are set to the same values specifed for AF_INET.

4 This additional NETWORK statement is required if you want a TCP/IP stack to also support IPv6. Omit this statement if you do not want the stack to support IPv6 (that is, the stack will support IPv4 only).

5 A transport provider stack for CINET is specified with a SUBFILESYSTYPE statement. The NAME field must match the address space name for the TCP/IP

started task as well as the TCPIPJOBNAME parameter in TCPIP.DATA. In our example, the name of the first stack is TCPIP1A and the name of the second stack is TCPIP1B.

6 EZBPFINI identifies a z/OS CS TCP/IP stack. For a z/OS CS TCP/IP stack, this is the only valid value.

7 Keyword DEFAULT specifies which transport provider stack is to be used as the default stack for z/OS UNIX. If DEFAULT is not specified, the first active stack will be used as the default stack. The sequence of SUBFILESYSTYPE statements is arbitrary if one stack is identified with the keyword DEFAULT. TCPIP1A is the default stack in Figure 7 on page 64.

Considerations for Enterprise Extender

The Enterprise Extender (EE) network connection is a simple set of extensions to the existing open high-performance routing (HPR) technology. It performs an efficient integration of the HPR frames using UDP/IP packets. To the HPR network, the IP backbone is a logical link. To the IP network, the SNA traffic is UDP datagrams that are routed without any hardware or software changes to the IP backbone. Unlike gateways, there is no protocol transformation and unlike common tunneling mechanisms, the integration is performed at the routing layers without the overhead of additional transport functions. The advanced technology enables efficient use of the intranet infrastructure for support of IP-based client accessing SNA-based data (for example, TN3270 emulators or Web browsers using services such as IBM's Host On-Demand) as well as SNA clients using any of the SNA LU types.

Enterprise Extender seamlessly routes packets through the network protocol *edges*, eliminating the need to perform costly protocol translation and the store-and-forward associated with transport-layer functions. Unlike Data Link Switching (DLSw), for example, there are no TCP retransmit buffers and timers and no congestion control logic in the router because it uses connectionless UDP and the congestion control is provided end system to end system. Because of these savings, the *edge* routers have less work to do and can perform the job they do best, which is forwarding packets instead of incurring protocol translation overhead and maintaining many TCP connections. Data center routers can handle larger networks and larger volumes of network traffic, thus providing more capacity. For more information, refer to the EE information in *Migrating Subarea Networks to an IP Infrastructure,* SG24–5957–00 (an IBM Redbook) or the following Web site:

`http://www-4.ibm.com/software/network/commserver/library/whitepapers/csos390.html`

Considerations for VIPA

The Internet Protocol (IP) is a connectionless protocol. IP packets are routed from the originator through a network of routers to the destination. All physical adapter devices in such a network, including those for client and server hosts, are identified by an IP Address which is unique within the network. The important point about IP is that a failure of an intermediate router node or adapter will not prevent a packet from moving from source to destination, as long as there is an alternate path through the network.

TCP sets up a connection between two endpoints, identified by the respective IP addresses and a port number on each. Unlike failures of an adapter in an intermediate node, if one of the endpoint adapters (or the link leading to it) fails, all connections through that adapter fail and must be reestablished. If the failure is on

a client workstation host, only the relatively few client connections are disrupted and usually only one person is inconvenienced. However, an adapter failure on a server means that hundreds or thousands of connections may be disrupted. On an S/390 or zSeries™ server with large capacity, the number may run to tens of thousands.

A Virtual IP Address, or VIPA in TCP/IP for z/OS , alleviates this situation. A VIPA is configured in the same way as a normal IP address for a physical adapter, except that it is not associated with any particular device. To an attached router, the TCP on z/OS simply looks like another router. When the TCP receives a packet destined for one of its VIPAs, the inbound IP function of the stack notes that the IP address of the packet is in the stack's Home list and passes the packet up the stack. Assuming the stack has multiple adapters or paths to it (including XCF from other TCP stacks in a sysplex), if a particular physical adapter fails, the attached routing network will simply route VIPA-targeted packets to the stack via an alternate route.

While this removes hardware and associated transmission media as a single point of failure for large numbers of connections, the connectivity of a server can still be lost through a failure of a single stack or an MVS image. The VIPA can be configured on another stack with a manual process, but this requires the presence of an operator or programmed automation.

Dynamic VIPA Takeover enables Dynamic VIPAs to be moved without human intervention or programmed automation to allow new connections to a server at the same IP address as soon as possible. This can reduce downtime significantly. With Dynamic VIPA Takeover you can configure one or more TCP/IP stacks to be backups (VIPABACKUP statement) for a particular Dynamic VIPA. If the stack or MVS image where the Dynamic VIPA is active is terminated, one of the backup stacks automatically activates that Dynamic VIPA. The existing connections will be terminated but can be quickly reestablished on the stack that is taking over.

Notes:

1. Because a VIPA is associated with a z/OS TCP/IP stack and is not associated with a specific physical network attachment, it can be moved to a stack on any image in the sysplex, or even to a z/OS TCP/IP stack not in the sysplex as long as the address fits into the installation's network configuration.
2. If using VIPA along with an intelligent bridge or switch, ensure that 'Port fast mode' (Cisco) is enabled. This helps to decrease the amount of time the VIPA is unreachable in scenarios where there is dynamic movement of VIPA (dynamic or static). For more information, see your bridge or switch manual.

You may also associate a particular Dynamic VIPA address with an application using the IOCTL SIOCSVIPA command or by BINDing explicitly to the Dynamic VIPA address. If the Dynamic VIPA address is within the VIPARANGE profile statement, then this Dynamic VIPA address will be created dynamically. This type of configuration enables a Dynamic VIPA to become an address of an application in a sysplex.

With Sysplex Distributor you can spread connection requests destined for Dynamic VIPAs to other stacks in the sysplex. You can use the VIPADISTRIBUTE profile statement to designate up to 32 stacks where connections for a particular DVIPA and up to 4 ports can be distributed, including the stack where the DVIPA is defined. The distributing stack (the stack where the VIPADISTRIBUTE statement was coded) might use either WLM or a combination of WLM and Quality of Service (QoS) performance information to determine where to forward new connection

requests. If the distributing stack/MVS image fails, connections forwarded to target stacks can be preserved by having the Dynamic VIPA address backed up on another stack.

Similarly, a stack can immediately take back a Dynamic VIPA address from another stack. If the original stack VIPADEFINEd the address with the keyword MOVEABLE IMMEDIATE (the default), then the Dynamic VIPA is moved as soon as the second stack requests ownership. The second stack assumes responsibility for forwarding packets for existing connections to the appropriate stack. If MOVEABLE WHENIDLE was specified, ownership does not pass until all existing connections on the current stack are closed.

For detailed information about VIPA, see Chapter 5, “Virtual IP Addressing” on page 209.

Required steps before starting TCP/IP

The following sections describe the steps you must complete before starting TCP/IP.

Planning your installation and migration

It will be to your advantage to have studied thoroughly the following documentation prior to the installation and customization of z/OS Communications Server:

- Program Directory for z/OS for CBPDO Installation and ServerPac Reference, Program Number 5694-A01
- Preventive Service Planning (PSP) bucket
- *z/OS Communications Server: IP Migration*
- *z/OS UNIX System Services Planning*
- OS390CKL, an IBM MKTTOOLS document for the z/OS UNIX System Services implementer

It is also recommended that you attend a z/OS UNIX System Services concepts class and a class in using z/OS UNIX System Services prior to migrating to z/OS Communications Server. If this is not possible, then you will want to ensure that the z/OS UNIX System Services implementer and the RACF administrator work together with you during the installation and customization process.

Planning for and installing z/OS Communications Server requires MVS, UNIX, and networking skill. If your background is in traditional MVS programming or systems programming, the z/OS UNIX System Services terminology might at first seem to be somewhat confusing. If your background is in the UNIX environment, the terms should be familiar to you.

In the past, MVS TCP/IP system programmers have needed a working knowledge of the MVS or z/OS system. These programmers have been accustomed to working closely with the RACF administrator and z/OS system programmer for authorizations; the VTAM and NCP system programmers for SNALINK and NCP connections; the IP address administrator for basic name and address assignments; and the administrators of the router network and channel-attached peripherals for connection definition and problem determination.

With the introduction of z/OS Communications Server, the TCP/IP system programmer needs to develop an additional alliance with the z/OS UNIX System Services system programmer. The TSO interfaces that have been traditionally available in the host-based TCP/IP still stand at the system programmer’s disposal

and additional MVS console commands simplify some TCP/IP operations. However, another user interface provided by the UNIX shell environment, either with the OMVS shell or the ISPF SHELL, is a useful and sometimes necessary tool that the TCP/IP system programmer will need to work with. Additionally, the tight coupling of z/OS Communications Server with z/OS UNIX System Services means that the TCP/IP system programmer needs more than a passing knowledge of UNIX conventions, commands, and Hierarchical File System (HFS) concepts. Even if the system programmer is familiar with other UNIX environments, work with the UNIX shell requires more than basic familiarity.

In the first version of a full TCP/IP stack based on native MVS and on z/OS UNIX System Services, few have all the requisite skills to successfully implement z/OS Communications Server on their own. As more and more systems programmers acquire skills in UNIX System Services and in TCP/IP, this will become less and less the case. Working with the z/OS UNIX System Services implementer when implementing z/OS Communications Server provides the most effective solution to establishing a working z/OS Communications Server environment.

Additional assistance is available to the z/OS UNIX System Services implementor at the z/OS wizards website, http:// www.ibm.com/eserver/zseries/zos/wizards/. Wizards are interactive assistants that simplify tasks such as installation planning, as well as configuration and customization.

If you are migrating to z/OS Communications Server, establish a migration process to move all your existing applications, and after this, consider the use of new and enhanced functions based on *z/OS Communications Server: IP Migration*. z/OS Communications Server allows multiple copies of the TCP/IP protocol stack to execute on the same MVS image. However, with all the performance enhancements introduced in z/OS Communications Server, it is probably not necessary to implement a multi-stack system for production purposes unless one is considering building a system programming test stack.

You are now ready to move on to the following steps.

Step 1: Install z/OS CS

Before you begin the installation:

- Read *z/OS and z/OS.e Planning for Installation* to help you plan the installation and migration of z/OS CS.
- Be sure you understand the data set naming conventions used in TCP/IP. You can find this information in “Configuration data set naming conventions” on page 19.
- Consult the *z/OS Program Directory* (Customization considerations for Wave 1D) for current information about the material, procedures, and storage estimates of the MVS image.

Install z/OS CS with other elements of z/OS. If you use the ServerPac method of installation, see *z/OS Installing Your Order*; if you use the CBPDO method of installation, refer to *z/OS Program Directory*. When appropriate, those two documents will direct you back to this document to customize the TCP/IP data sets and procedures and verify their configuration.

Verifying the initial installation

Both the *z/OS Program Directory* and *z/OS Installing Your Order* contain step-by-step instructions that can be used to set up and verify a basic TCP/IP

configuration with only the loopback address and a few key servers. For more information regarding these instructions, refer to the information about Wave 1D customizations in the *z/OS Program Directory* or the information about verifying your installation in *z/OS Installing Your Order*.

Step 2: Customize z/OS CS

To customize TCP/IP you need to update the cataloged procedures and configuration data sets for the TCP/IP address space, its clients, and servers.

z/OS CS runs as a started task in its own address space. Each of the servers runs in its own address space and is started with its own procedure. The TCP/IP address space requires:

- A procedure in a system or recognized PROCLIB.
- A data set that provides configuration definitions for the TCP/IP address space and includes statements affecting many of the servers. This data set is referred to as PROFILE.TCPIP.
- A data set to provide the parameters that are common across all clients. This data set is referred to as TCPIP.DATA.

Many of the servers also require other data sets for their specific functions.

Making SYS1.PARMLIB changes

You need to make certain changes to SYS1.PARMLIB. These changes depend on which of the following installation methods you use:

ServerPac method

After the file system is restored (through the RESTFS job), you will see that ServerPac has changed some of the PARMLIB members. Follow the instructions to change the BPXPRMxx member of PARMLIB.

CBPDO method

Change the PARMLIB members according to the instructions listed in the chapters that describe installation instructions for Wave 1. Tables describing changes to PARMLIB and changes to BPXPRMxx member are included.

Note:

z/OS CS exploits z/OS UNIX services even for traditional MVS environments and applications. Prior to utilizing TCP/IP services, therefore, a full-function mode z/OS UNIX environment—including a Data Facility Storage Management Subsystem (DFSMSdfp™), a Hierarchical File System (HFS), and a security product (such as Resource Access Control Facility (RACF))—needs to be defined and active before z/OS CS can be started successfully.

Additional information about required TCP/IP definitions for the UNIX environment can be found in “Defining TCP/IP as a UNIX System Services physical file system (PFS)” on page 50 and “UNIX System Services security considerations” on page 45.

Common z/OS UNIX configuration problems: Following are some explanations and possible solutions for common problems that you may encounter when configuring the z/OS UNIX environment.

- TCP/IP initialization fails with the following messages:

```
EZZ4203I OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIP-BPX1SOC,
     00000003,FFFFFFFF,00000070,112B00B6
```

These messages usually indicate that both INET and CINET FILESYSTYPE have been specified. Only one should be specified; refer to the FILESYSTYPE section in *z/OS UNIX System Services Planning* for additional information.

- TCP/IP initialization fails with the following messages:

```
EZZ4203I OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIP-BPX1SOC,
     00000003,FFFFFFFF,0000006F,112B00B0
```

These messages indicate that the requester of the service is not privileged. The service requested requires a privileged user. Check the documentation for the service to understand what privilege is required.

- TCP/IP initialization fails with the following messages:

```
EZZ4203I OPENEDITION-TCP/IP CONNECTION ERROR
     FOR TCPIPA-BPX1IOC,8008C981,FFFFFFFF,0000009E,12B2005A

EZZ4204I TCPIP INITIALIZATION FOR TCPIPA FAILED
```

These messages usually indicate that an incorrect jobname was specified in the SUBFILESYSTYPE NAME() definition in the BPXPRM*xx* member for a common INET environment. In this scenario, the NAME() must match TCPIPA.

- TCP/IP initialization fails with the following messages:

```
IEA8481 DUMP SUPPRESSED - ABDUMP MAY NOT DUMP STORAG FOR KEY 0-7 JOB TCPV34A
IEF4501 TCPIPA TCPIPA - ABEND=SEC6 U0000 REASON=0F01C008
```

These messages are usually an indicator that an OMVS RACF segment has not been defined for the user ID associated with the TCP/IP started procedure. Define an OMVS segment with a UID of 0 for the user ID associated with the TCP/IP started procedure.

- TCP/IP initialization fails with the following messages:

```
IEF4031 TCPIPA - STARTED - TIME=16.01.25
EZZ42031 OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIPA-BPX11OC,
     8008139A,FFFFFFFF,00000079,12D2025E
EZZ42041 TCPIP INITIALIZATION FOR TCPIPA FAILED.

==> The 0079 value is EINVAL - The parameter is incorrect
==> The 025E value is JRSocketCallParmError - A socket syscall
    contains incorrect parameters
```

These messages usually indicate that an incorrect entry point name has been specified in the SUBFILESYSTYPE ENTRYPOINT() definition. The correct value is ENTRYPOINT(EZBPFINI).

- TCP/IP initialization fails with the following messages:

```
EZZ32031 OPENEDITION-TCP/IP CONNECTION ERROR FOR TCPIPA-BPX1SOC,
     00000003,FFFFFFFF,0000045A,112B0000
EZZ4204I TCPIP INITIALIZATION FOR TCPIPA FAILED.

==> The 045A value is EAFNOSUPPORT - The address family is not supported
```

These messages indicate that AF_INET was not defined or did not initialize properly. Check for any earlier z/OS UNIX messages and verify that the z/OS UNIX NETWORK DOMAINNAME(AF_INET) statement is in your BPXPRM*xx* member.

- After issuing a NETSTAT command from TSO, the following message is displayed:

```
netstat
CEE5101C During initialization, the z/OS UNIX callable service
     BPX1MSS failed. The system return code was 0000000156,
     the reason code was 0507014D. The application will be
```

```
     terminated.
NETSTAT ENDED DUE TO ERROR+
READY
?
USER ABEND CODE 4093 REASON CODE 00000090
READY

==> The 0156 value is EMVSINITIAL - Process initialization error
==> The 014D value is JRFsFailChdir - The dub failed, due to
    an error with the initial home directory
```

These messages indicate that the user ID issuing the NETSTAT command does not have an OMVS RACF segment defined for it. Define an OMVS segment for this user ID or activate the default OMVS segment support. For details, see "UNIX System Services security considerations" on page 45.

- Socket applications using the z/OS CS TCP/IP Services APIs fail with an ERRNO of 156.

  ERRNO 156 indicates a z/OS UNIX process initialization failure. This is usually an indication that a proper OMVS RACF segment is not defined for the user ID associated with the application. The RACF OMVS segment may not be defined or may contain errors such as an improper HOME() directory specification. If the OMVS segment is not defined, you may also receive the following message:

  ```
  ICH4081 USER(USER8   ) GROUP(SYS1   )   NAME(TSO USERID USER8  )
     CL(PROCESS )
     OMVS SEGMENT NOT DEFINED
  ```

  In this example, USER8 is the user ID associated with the failing application. To correct this problem, define a proper OMVS segment for the user ID associated with the failing application. For details, see "UNIX System Services security considerations" on page 45.

Completion of these steps ensures that the applications and resources on the target system will function correctly at the new level.

The subsequent chapters in this document show you how to:

- Configure the TCP/IP address space by updating the samples provided in *hlq*.SEZAINST(SAMPPROF) and *hlq*.SEZAINST(TCPIPROC).
- Configure the universal client parameters provided in *hlq*.SEZAINST(TCPDATA).
- Configure the site table, defined in *hlq*.HOSTS.LOCAL or *hlq*.ETC.IPNODES, to identify the Internet names and addresses of your TCP/IP host.
- Customize the TCP/IP Component Trace parameters by updating the CTRACE parameter in the PARM= field of the EXEC JCL statement in the TCP/IP started procedure.

  You can find a description of the MVS Component Trace support in the *z/OS Communications Server: IP Diagnosis*.
- Specify the ENVAR parameter on the PARM=keyword to override the resolver file. For more information on setting the environment variable RESOLVER_CONFIG using the ENVAR parameter, see "Considerations for multiple instances of TCP/IP" on page 54.
- Configure each of the servers you want to run. This might require:
  - Modifying sample procedures and adding them in your PROCLIB
  - Modifying the configuration data set, PROFILE.TCPIP
  - Adding port numbers to *hlq*.ETC.SERVICES
  - Modifying other data sets containing server-specific parameters

You can find the sample procedures and data sets in *hlq*.SEZAINST or the HFS. Table 2 on page 21 provides additional reference information you can use as you configure and customize each server.

You can find general information about starting, stopping, and dynamically controlling the servers in *z/OS Communications Server: IP System Administrator's Commands*.

Step 3: Configure VMCF and TNF

The Pascal socket interface makes use of the IUCV/VMCF services for a limited set of inter-address space communication flows. As a result, if you are using any applications (provided by IBM or others) that use the Pascal socket API, you must insure that the VMCF and TNF subsystems are active before the applications are started. TCP/IP provides several applications and commands that exploit these interfaces, such as the SMTP and LPD servers, and the TSO REXEC, RSH, and remote printing commands; therefore, almost all installations will require setting up VMCF and TNF.

The restartable VMCF must be started before TCP/IP if you want the VMCF node name used as a default host name during TCP/IP initialization (in cases where no other host name can be located).

Note: Host name is the value normally specified on the TCPIP.DATA HOSTNAME statement.

Also note that the VMCF node name is used as a system name qualifier when processing the TCPIP.DATA file and by the SMTP server as the NJE node name. It is recommended that the MVS system name is used for the VMCF node name specification and that the NJE node name is specified explicitly by using the NJENODENAME statement in the SMTP configuration data set.

You can configure Virtual Machine Communication Facility (VMCF) and TNF in two different ways: as restartable subsystems or as non-restartable subsystems.

Restartable subsystems

Configuring VMCF and TNF as restartable subsystems has the following advantages:

- Error detection is provided when the subsystems do not seem to be initializing properly.
- You can change the system name on the restart.
- Commands are available to remove users from internal tables, display current users and to terminate the subsystem.

In summary, a restartable VMCF and TNF configuration provides better availability and is therefore recommended.

If you choose to use restartable VMCF and TNF, follow these steps:

1. Update your IEFSSN*xx* member in SYS1.PARMLIB with the TNF and VMCF subsystem statements required by TCP/IP. The specification can be in either the IBM recommended keyword parameter form or the positional parameter form of IEFSSNxx. For example:

```
* The keyword parameter form is:

SUBSYS SUBNAME(TNF)
SUBSYS SUBNAME(VMCF)
```

```
* The positional parameter form is:
TNF
VMCF
```

2. Add procedure EZAZSSI to your system PROCLIB. A sample of this procedure is located in the data set *hlq*.SEZAINST (where *hlq* is the high-level qualifier for the TCP/IP product data sets in your installation).

   ```
   //EZAZSSI PROC P=&SYSNAME.
   //STARTVT EXEC PGM=EZAZSSI,PARM=&P,TIME=1440
   ```

3. Start VMCF and TNF using the procedure EZAZSSI before starting TCP/IP. If your nodename is the same as the MVS system symbolic &SYSNAME, then you can start VMCF and TNF with the following command:

   ```
   S EZAZSSI
   ```

   If your nodename is different than the MVS system symbolic &SYSNAME, start VMCF and TNF as follows:

   ```
   S EZAZSSI,P=nodename
   ```

Replace nodename with the SYSTEM NAME of your MVS system.

Non-restartable subsystems

If you will not be using restartable VMCF and TNF, you should update your IEFSSN*xx* member in SYS1.PARMLIB with the following subsystem statements required by TCP/IP. The specification can be in either the IBM recommended keyword parameter form or the positional parameter form of IEFSSNxx. For example:

```
* The keyword parameter form is:

SUBSYS SUBNAME(TNF) INITRTN(MVPTSSI)
SUBSYS SUBNAME(VMCF) INITRTN(MVPXSSI) INITPARM(nodename)


* The positional parameter form is:

TNF,MVPTSSI
VMCF,MVPXSSI,nodename
```

Do not use the sample SEZAINST (IEFSSN) as shipped, because the comments are not valid in SYS1.PARMLIB. A modified form of the last two lines must be placed in the IEFSSN*xx* PARMLIB member. Replace node name on the VMCF line with the NJE node name of your MVS system.

VMCF commands

If you will be using restartable VMCF, the following VMCF commands let you display the names of the current users of VMCF and TNF, and if necessary, remove names from the name lists.

Note: Removing names from the name lists and stopping either subsystem can have unpredictable results, if done hastily. Use the REMOVE and stop (P) commands carefully and only as a last resort.

If you remove a user, the application is not canceled, nor is the connection severed. In other words, the *removed* application may remain active in the system, and may subsequently abend 0D6/0D4/0C4, or cause TCP/IP to hang. A user that is removed from VMCF may still be a user of TNF and even TCP/IP, and vice versa.

To terminate users and stop VMCF or TNF properly, follow these steps:

1. Display the current users of the subsystems, using one of the following:

   ```
   F VMCF,DISPLAY,NAME=*
   F TNF,DISPLAY,NAME=*
   ```

2. Terminate those users. If termination fails, use the REMOVE command as a last resort to force them from the name list.
3. Stop the subsystem, using one of the following commands:

   ```
   P VMCF
   P TNF
   ```

   If the P command fails, use one of the following commands:

   ```
   FORCE ARM VMCF
   FORCE ARM TNF
   ```

Following are descriptions of the commands:

F TNF,DISPLAY,NAME=[name|*]
: Displays the named user [or all (*) users] of TNF, sorted by ASID.

F TNF,REMOVE,NAME=[name|*]
: Removes either the named user [or all (*) users] from the TNF internal tables.

P TNF Requests TNF to terminate.

F VMCF,DISPLAY,NAME=[name|*]
: Displays the named user [or all (*) users] of VMCF, sorted by name.

F VMCF,REMOVE,NAME=[name|*]
: Removes either the named user [or all (*) users] from the VMCF internal tables.

P VMCF
: Requests VMCF to terminate

Following are sample commands:

```
F TNF,DISPLAY,NAME=TCPV3
F VMCF,DISPLAY,NAME=*
F TNF,REMOVE,NAME=FTPSERV
F VMCF,REMOVE,NAME=*
P TNF
```

Common VMCF problems

Following are some common VMCF problems:

- VMCF or TNF fail to initialize with an 0C4 abend.

  This is probably an installation problem; check the PPT entries for errors. Some levels of MVS do not flag PPT syntax errors properly.
- Abends 0D5 and 0D6 after REMOVEing a user.

  This is probably because the application is still running and using VMCF. It is not recommended that users be removed from VMCF or TNF without first terminating the affected user.
- VMCF or TNF do not respond to commands.

  This is probably because one or both of the non-restartable versions of VMCF or TNF are still active. To get them to respond to commands, stop all VMCF/TNF users, FORCE ARM VMCF and TNF, then use EZAZSSI to restart.
- VMCF or TNF cannot be stopped.

This is probably because users still exist in the VMCF and TNF lists. Use the *F VMCF,DISPLAY,NAME=** and *F TNF,DISPLAY,NAME=** commands to identify those users who are still active. Then either cancel those users or remove them from the lists using the *F VMCF,REMOVE* and *F TNF,REMOVE* commands.

IUCV/VMCF considerations

The IUCV/VMCF inter-address space communication API enables applications running in the same MVS image to communicate with each other without requiring the services of the TCP/IP protocol stack. The VMCF/TNF subsystems provide these services, which are still available in z/OS CS. Several components of TCP/IP in z/OS CS continue to make some use of these services for the purpose of inter-address space communications. These include:

- The AF_IUCV domain sockets for the TCP/IP C socket interface. The AF_IUCV domain enables applications executing in the same z/OS image and using the TCP/IP C socket interface to communicate with each other using a socket API, but without requiring the services of the TCP/IP protocol stack, as no network flows result in these communications. This is quite different from the more common AF_INET domain that enables socket communication over a TCP/IP network. AF_IUCV sockets continue to be supported in z/OS CS.

  An example of a TCP/IP-provided application that exploits AF_IUCV sockets is the SNMP Query Engine component (SQESERVE). The z/OS UNIX socket library provides a similar functionality to the AF_IUCV domain sockets with its AF_UNIX domain. Users creating new applications should consider using AF_UNIX domain sockets.
- The Pascal socket interface also makes use of the IUCV/VMCF services for a limited set of inter-address space communication flows. As a result, any applications (provided by IBM or others) that use the Pascal socket API also still have a requirement for the VMCF/TNF subsystems. TCP/IP provides several applications and commands that exploit these interfaces, such as the SMTP and LPD servers, and the TSO TELNET, HOMETEST, TESTSITE, RSH, REXEC, and LPR commands.

Therefore, in z/OS CS you must continue to configure and start the VMCF and TNF subsystems as you did in TCP/IP V3R2. However, because the VMCF/TNF subsystems are no longer used to communicate directly with the TCP/IP protocol stack in z/OS CS, the amount of CPU they will consume will be significantly lower than in the TCP/IP V3R2 environment.

Step 4: Update the VTAM application definitions

You must update the VTAM definitions for TN3270 Telnet and any other of these applications that you configure on your system. You can find example VTAM definitions for each of these applications in their respective chapters.

- SNALINK
- SNALINK LU6.2
- TN3270 Telnet
- X.25 NPSI Server

hlq.SEZAINST(VTAMLST) contains a sample of the VTAM definitions for TN3270 Telnet applications. You should copy this member, update it, and add it to the ATCCON*xx* member of VTAMLST. This will ensure that the TN3270 Telnet applications are activated when VTAM is started.

Because the TCP/IP LU code cannot handle multiple concurrent sessions, you must code SESSLIM=YES for each TN3270 Telnet LU defined to VTAM. Otherwise, if

SESSLIM=NO, menu or session manager applications that use return session processing might cause session termination.

Step 5: Verify that the resolver address space is active

The resolver address space must be started before the TCP/IP address space can be started. For information on how the resolver can be started, see "Understanding resolvers" on page 12. You can use the resolver's MODIFY DISPLAY command to check that the resolver is active and what resolver setup statements are being used. For the syntax and usage of the command, see *z/OS Communications Server: IP System Administrator's Commands*.

Step 6: Start the TCP/IP address space

Enter the MVS START command from the operator's console to start TCP/IP, specifying the member name of your cataloged procedure. This will start the TCP/IP address space and any of the servers you have defined in the AUTOLOG statement in PROFILE.TCPIP. For example, if the procedure to start the TCP/IP address space was called TCP1 in your PROCLIB, you would enter:

```
START TCP1
```

For information on updating the TCPIP cataloged procedure or configuration statements used to configure the TCPIP address space, refer to *z/OS Communications Server: IP Configuration Reference*.

Step 7: Set up cataloged procedures and configuration data sets

At this point in the configuration process, you can choose to either set up procedures or you can do each one individually when you set up the appropriate application, function, or server.

See the remaining chapters in this document for more information about setting up the appropriate application, function, or server.

Step 8: Customize TCP/IP messages

The messages for every TCP/IP server program are compiled and linked with the program and reside in an internal message repository. Some of the server programs that are written in the C language also have their messages in external data sets. You can edit these external message data sets to translate the messages to another language or customize them to suit your installation.

How to access the message data sets

The procedures for these servers have a special DD statement that point to the external message data set. If you are going to override the internal messages and use external customized messages, you need to remove the comment from the appropriate DD statement and ensure it points to the correct data set.

The following table shows the servers that have external messages, the DD statement used, and the name of the message data set delivered with the system:

Server	DD statement	Data set
NCPROUTE	//MESSAGE	SEZAINST(EZBNRMSG)
SNMP Query Engine	//MSSNMPMS	SEZAINST(MSSNMP)
MISC Server	//MSMISCSR	SEZAINST(MSMISCSR)

Message text

The message text might include special characters for the variable fields that are converted when the message is printed or displayed and control characters that affect the message format. The conversion characters start with a percent sign (%) and the control characters start with a backslash (\). These are all standard notations for the C language print function. The messages might also contain comments which start with `/*` and end with `*/`.

In the following simulated message, the control character `\n` forces a new line to print and the string variables, represented by `%s`, are converted in the order they are passed from the program.

```
29999  I  Command %s received from user %s\n
```

Message format

The following diagram explains the syntax for TCP/IP message IDs on the host:

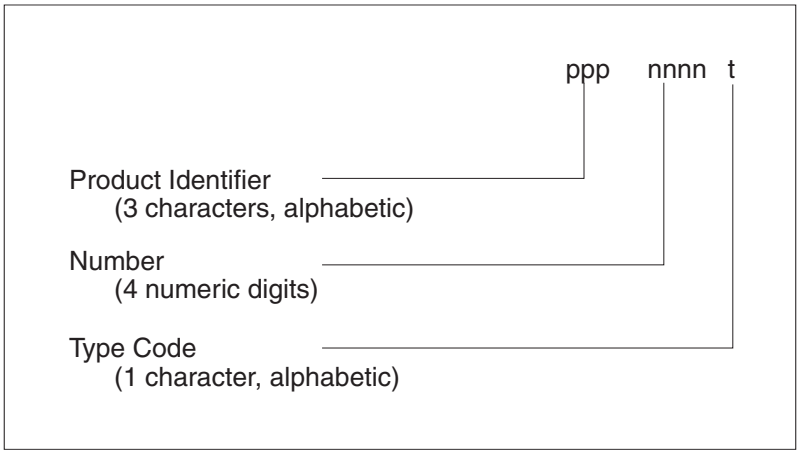


Figure 8. Syntax for TCP/IP message IDs

The ***product identifiers*** (ppp) for TCP/IP are EZA, EZB, EZY, and EZZ. The ***number*** (nnnn) indicates a unique 4-digit numeric value assigned to the message by product. The ***type*** (t) indicates the severity assigned to the message.

Rules for customizing the messages

The general rule for customizing or translating messages is to only change the text portion of the message.

- Do not change the MARGIN, PRODUCT, and COMPONENT definitions at the top of the data set. These are required definitions for the program. For example, these entries at the top of the MISC server message data set should not be changed:

  ```
  MARGINS(1,72)
  PRODUCT EZA
  COMPONENT MSC
  ```

- Do not change the message numbers and the severity code. These parts of the message have specific meaning; if you change them the program may not work correctly.
- Do not change the conversion characters. These indicate that the program is passing data, the type of data it is passing, and the appropriate way to display or print this data. For example, do not change or delete %s and %d in the following message:

```
4858 W   "Route from %s in unsupported address family %d\n"
```

- You can reorder the variables that are passed in the message. For example, you can reverse the order of the two string variables that are passed when translating a message by specifying the new order of the arguments in parentheses following the message text:

  Before: `29999I  Command %s received from user %s\n`

  After: `29999I  Utilisador %s envio instrucion %s\n (2, 1)`

  The result would be `EZY9999I Utilisador MANNY envio instrucion FTP` instead of `EZY9999I Command FTP received from user MANNY`.

- Watch for any program parameters or keywords that might be in the message text. In most cases, you should not translate them.

  For example, in the following message, `'active'` is a keyword used in the gateway definition and should not be translated:

  ```
  4851 E   "First two elements must be 'active' for active gateway\n"
  ```

Chapter 2. Security

The z/OS Communications Server, along with other elements of z/OS, provide numerous enterprise-strength security services to protect your mission-critical data. This chapter provides an overview of these technologies and how they can be used for a safe and secure z/OS TCP/IP deployment.

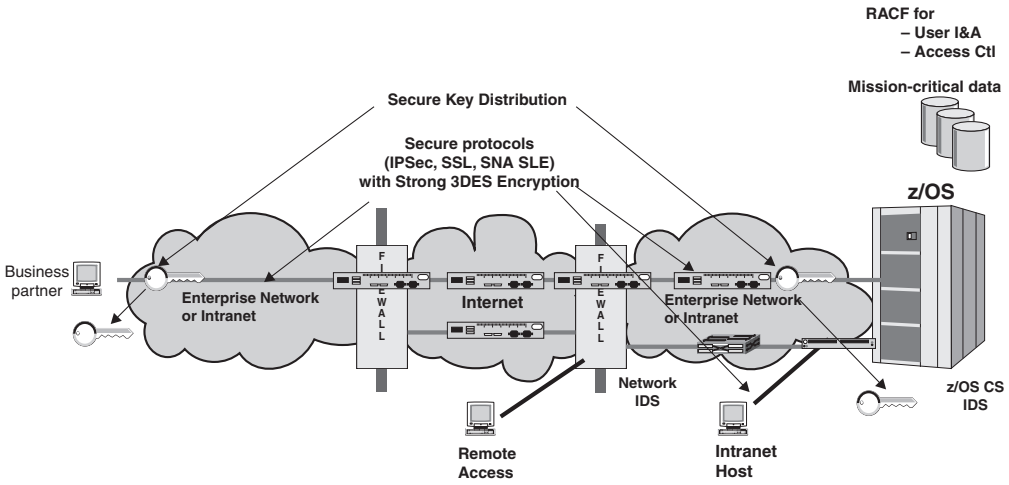


Figure 9. Elements of a secure TCP/IP deployment

The Communications Server protects data and other resources on the system. Communications Server applications use RACF services to ensure that users requesting application access are identified and authenticated, and to protect data and other system resources from unauthorized access. The Communications Server safeguards the availability of the system by protecting against denial of service attacks from the network.

The Communications Server protects data in the network by supporting a variety of cryptographic-based network security protocols such as IPSec, SSL, and SNA Session Level Encryption. These security protocols ensure that data received is originated by the claimed sender (data origin authentication), that contents were unchanged in transit (message integrity), and that sensitive data is concealed using encryption (data privacy).

The Communications Server provides security event reporting to record potential security violations. These services may help you identify potential sources of subsequent attacks, respond more quickly to network attacks, and manage system resources during periods of high network traffic for key applications.

Note: Some of the security features described in this chapter have not yet been implemented for IPv6. To determine which functions are supported for IPv6, see Table 1 on page 3.

System resource protection

Application security

The Communications Server protects data and other system resources accessed by applications included in the Communications Server element. This protection

requires verification of the identity of the end user requesting access. This process is called identification and authentication. In addition, access to resources must be limited to those users with permission. This process is called access control. Communications Server applications use RACF for identification and authentication, and access control decisions. Authenticated users are granted access to RACF resources only for which they have permission

Some applications allow anonymous access. Applications that allow anonymous access include anonymous FTP, Remote Execution, and Trivial File Transfer Program (TFTP). The Communications Server ensures that all anonymous access can be controlled by the installation. If anonymous access is allowed, the resources accessed can be limited in several ways:

- The application can be configured to limit resources for which access will be attempted.
- The application can be configured to use a RACF user ID to represent the anonymous user. In this case, access is allowed for those resources specifically permitted for the anonymous RACF user ID and for those resources that are universally accessible.

Most Communications Server applications must be configured specifically to allow anonymous access. One exception is TFTP. TFTP allows anonymous read access only. TFTP can be configured to control those directories that contain files that can be downloaded.

The following chart depicts a representative set of Communications Server applications, whether end user identification is required, and the security credentials under which resource access is made. For more information on specific application considerations, refer to the individual chapters for each application.

Server	End User Identification	Resource Access
FTP	Optional (1)	End user ID or configured anonymous user ID (2)
LPD	Optional (1)	Server ID or end user ID
TFTP	No	Server user ID (2)
MVS REXECD	Required	End user ID
MVS RSHD	Required (password optional) (1)	Surrogate user ID or end user ID
UNIX REXECD	Required	End user ID
UNIX RSHD	Required (password optional) (1)	End user ID or Server user ID(exit routine to verify request)
UNIX SHELL (telnet/rlogin)	Required	End user ID

Figure 10. User identification, authentication, and access control for z/OS Communications Server applications

(1) All optionals are installation controlled and can all be configured to require full end user identification.

(2) Files accessible can be configured on a server basis to limit access.

TCP/IP resource protection

The Communications Server uses the System Authorization Facility (SAF) to protect TCP/IP resources from unauthorized access. These resources are represented by resource profiles defined in the SERVAUTH class. The use of SERVAUTH is optional. The installation can choose to use any combination of the protections provided by SERVAUTH.

In addition to the use of SERVAUTH protection, other functions provide further resource protection such as Intrusion Detection Services (IDS), syslogd isolation and IP filtering. These topics are discussed in more detail later in the chapter.

Local user access control to TCP/IP resources using the SAF

The SAF can control the ability of users executing on z/OS to access select TCP/IP resources. These functions protect against unauthorized user access to:

- The TCP/IP stack
- TCP and UDP ports
- The IP network or specific hosts in an IP network
- Netstat command output
- Webserver page caching services in the TCP/IP stack

With this solution, the administrator defines the above TCP/IP resources as SAF resources. The resource profiles are defined as part of the SERVAUTH class. The Communications Server allows the local user access to these resources based on the user or group permissions associated with the SAF resource.

Stack Access Control

Stack Access Control allows control of access to a TCP/IP stack using the SAF. It provides a way to generally allow or disallow users or groups of users access to a TCP/IP stack. The function controls the ability of a user to open an AF_INET socket. The TCP/IP stack to be protected is represented with a SERVAUTH profile name *EZB.STACKACCESS.sysname.tcpname*. Access to the stack is allowed if the user is permitted to this resource. There are no new TCP definitions required. The function is enabled if the SERVAUTH class is active and the stack access resource is defined. If it is not defined, the stack access check is not made.

Note: Some security products do not distinguish between a resource profile not defined and a user not permitted to that resource. If your product does not make this distinction, then you must define the stack access resource profile and permit users to it whenever the SERVAUTH class is active.

The following example provides an overview of Stack Access Control. *sysname* refers to the MVS system variable *sysname*. *tcpname* refers to the TCP/IP job name. As shown in the example below, user Tom has permission to access both Stack1 and Stack2, Joe does not have permission to access any stack, and Bob has permission to access Stack2 but not Stack1.

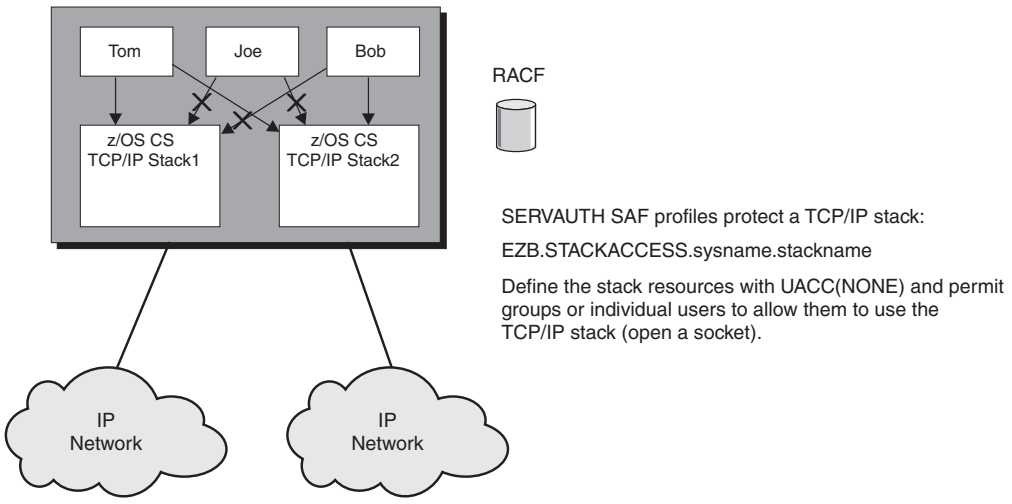


Figure 11. Stack Access Control overview

Port Access Control

Port Access Control uses the PORT and PORTRANGE statements to protect against unauthorized use of non-ephemeral ports. It allows control of an application's ability to bind to specific TCP and UDP ports or port ranges using the SAF. The port access support is enabled if the keyword, SAF, is specified on the PORT or PORTRANGE statement. The SAF keyword value specifies a portion of the resource name that represents the port. The user ID associated with the application at the time of the bind request must be permitted to the resource before the application is allowed to bind to the port. The port is represented by a SERVAUTH profile name of *EZB.PORTACCESS.sysname.tcpname.SAFkeyword*. *SAFkeyword* is the value specified on the SAF keyword on the PORT and PORTRANGE statement.

The following example provides an overview of Port Access Control. As shown in the example below, z/OS user WEBSERV is permitted to bind to port 80. User Bob is not permitted to bind to port 80.

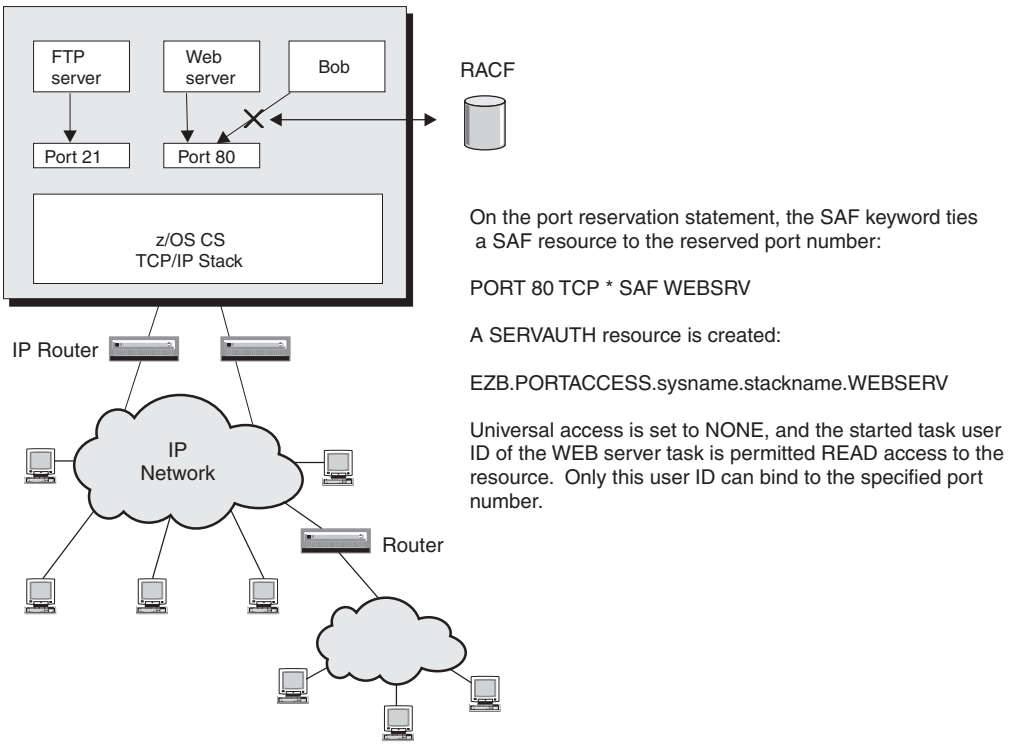


Figure 12. Port Access Control overview

Port Access Control also augments the job name reservation method. The PORT or PORTRANGE may be reserved with a job name, a wildcard job name (*), or the special job name of RESERVED. If job name is specified, the port is reserved for an application with the specified job name. If the wildcard job name is specified, the port is not reserved for any particular job name. For both of these cases, the SAF keyword, if specified, still verifies that the userid associated with the application at the time of the bind to the port is permitted access to the port. The RESERVED job name shuts down the use of a port or range of ports for any application.

The IPCONFIG, UDPCONFIG, and TCPCONFIG RESTRICTLOWPORTS statements specify that all applications binding to a low port (1–1024) must be APF-authorized or superuser, unless the SAF keyword is specified and the user ID binding to the port is permitted to the SAF resource. z/OS CS client applications that need to bind to a low port are shipped as APF-authorized.

Network Access Control

Network Access Control gives system administrators the ability to assign permission for z/OS users to access certain networks and hosts. With this function the ability of users to send or receive data between z/OS and certain networks can be controlled at the z/OS. Network Access Control provides an additional layer of security to any authentication and authorization security that is used in the network or at the peer system by disallowing the unauthorized user to communicate with the peer network resource.

Essential elements of this function are as follows:

- The IP network is considered the resource to be protected.
- IP addresses are classified into *security zones*, in which each zone has a certain level of security sensitivity. A default security zone exists for interfaces that are not explicitly associated with a specific security zone. Security zones consist of

one or more, perhaps discontiguous, IP address ranges that have the same security sensitivity and are identified by a specific zone name.

- The SAF is used to check permission of users or groups of users to access the security zone.
- The installation defines a network access resource for each security zone and permits users or groups of users access to the resource. The security zone is represented by an SAF SERVAUTH profile name of *EZB.NETACCESS.sysname.tcpname.zonename*.
- TCP/IP keeps a mapping of network resources by IP address to security zones. This mapping is consulted on certain inbound and outbound operations to determine the corresponding resource zone name for the most specific network defined. Then the current user's access to that resource is queried using the SAF, and the operation will be allowed or denied completion accordingly. This mapping is also consulted when the security ioctl is issued to extract the port of entry zone name of a socket's current peer.
- Network Access Control is used to control z/OS user access to an IP network via a sockets application. Resource access checks will occur when an application explicitly binds a socket to a local address, including the address INADDRANY (0.0.0.0/32). Resource access checks will occur at connection setup or acceptance time for TCP, peer identification time for UDP and RAW, and on the first and potentially subsequent sends or receives (TCP, UDP, or RAW) to a particular destination in a socket's lifetime. Additionally, there is no user concept when dealing with packets that are being forwarded through the stack and hence no checks will be made. Network Access Control security checks are made at the transport layer (TCP, UDP, and RAW). Other IP specific packets generated by the stack are not covered under this function (such as ICMP echo replies, for example).
- Network Access Control for outbound and inbound can be individually enabled or disabled.
- TCP/IP caches security information following Network Access Control checks. The NetAccess zone table in the TCPIP PROFILE must be rebuilt to cause TCP/IP to recognize changes to the SERVAUTH class profiles for existing sockets.

The following example provides an overview of Network Access Control. As shown in the example below, z/OS user Bob is permitted access to Security Zone A but not Security Zone B. An outbound connect from Bob is permitted to Security Zone A, but not Security Zone B. Bob is permitted to accept connections from Security Zone A but not Security Zone B.

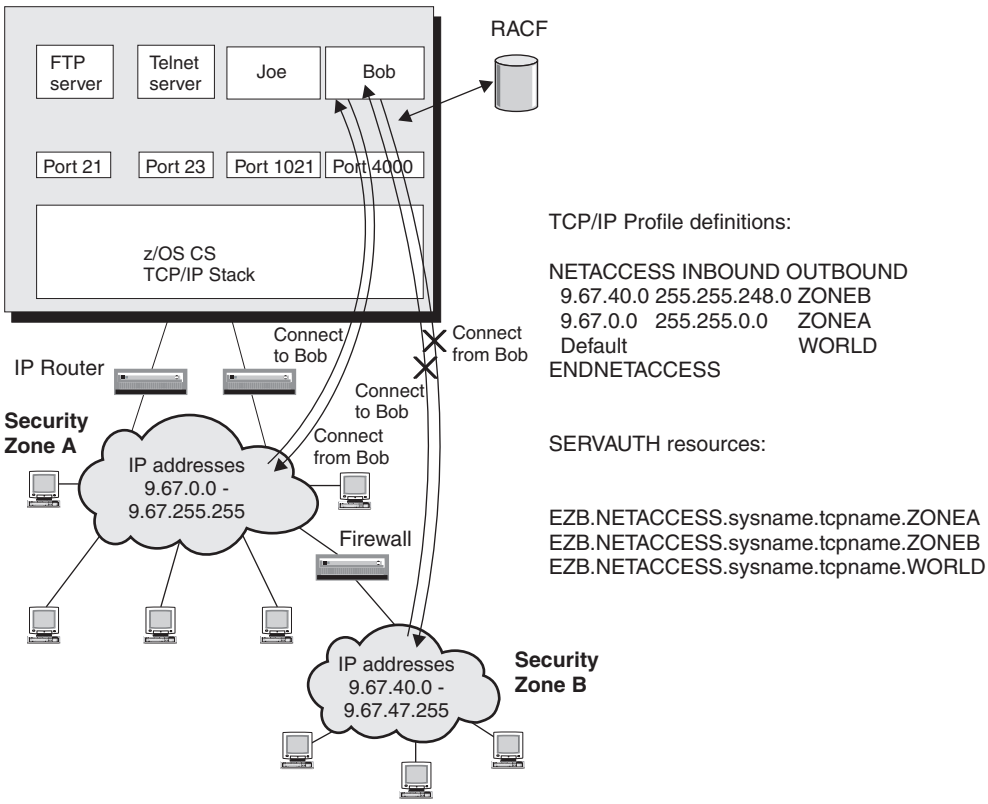


Figure 13. Network Access Control example

Netstat Access Control

Netstat Access Control allows control of access to Netstat command output from the TSO or UNIX System Services shell environments using the SAF. The Netstat command output is considered the resource to be protected and is represented with a resource profile in the SERVAUTH class named *EZB.NETSTAT.sysname.tcpname.netstatoption*. Access to the Netstat output is allowed if the user is permitted to this resource. There are no new TCP definitions required. The function is enabled if the SERVAUTH class is active and the netstat option resource is defined. If it is not defined, the check is not made.

Note: Some security products do not distinguish between a resource profile not defined and a user not permitted to that resource. If your product does not make this distinction, then you must define the netstat resource profiles and permit users to them whenever the SERVAUTH class is active.

An installation can implement a security policy that indicates which users have authorization to selected Netstat options. The level of granularity for this security policy can be either by individual or all Netstat options.

Fast Response Cache Accelerator Access Control

Fast Response Cache Accelerator Access Control allows control of application access to Fast Response Cache Accelerator (FRCA) services. The FRCA configuration ioctl is considered the resource to be protected and is represented with a resource profile in the SERVAUTH class named *EZB.FRCAACCESS.sysname.tcpname*. Access to FRCA services is allowed if the Web server user is permitted to this resource. There are no new TCP definitions

required. The function is enabled if the SERVAUTH class is active and the FRCA access resource is defined. If it is not defined, the check is not made.

Note: Some security products do not distinguish between a resource profile not defined and a user not permitted to that resource. If your product does not make this distinction, then you must define the FRCA access resource profile and permit users to it whenever the SERVAUTH class is active.

Syslogd isolation

Syslogd isolation provides a capability for the installation to control which user IDs and job names can write syslogd records to specified syslogd facilities. This function enables the installation to segregate system and application syslogd records, and to segregate syslogd records from different applications. This function prevents an application level process from flooding a syslogd facility intended for system use, possibly causing system syslogd records to be lost. This function is enabled when user ID and/or job name are specified as additional criteria along with existing facility and priority criteria to select a syslogd repository.

In addition, the user ID and job name associated with the syslogd record writer can optionally be stored in a syslogd record based on a syslogd command-line parameter. This capability is useful when syslogd records for multiple jobs or users are recording in the same syslogd facility. This function enables positive identification of the creator of the syslogd records and ensures that the syslogd record, if spoofed, can be identified.

Syslogd isolation also provides a capability to disable reception of syslogd messages from other hosts in the network. This capability is provided by a syslogd command-line parameter. This parameter disables reception of syslogd messages from all hosts. If an installation wants to allow certain hosts in the network access to syslogd, IP Filtering can be used instead to specify which hosts are permitted to access the syslogd UDP port.

IP filtering

The Security Server can configure the Communications Server to perform packet filtering at the IP layer. IP filters are rules defined to either discard or permit packets. IP filtering matches a filter rule to data traffic based on any combination of IP source or destination address (or masked address), protocol, source or destination port, direction of flow, or time. IP filtering can control traffic being routed, or control access at the host that has the communication endpoint. Even when an external firewall is providing filtering protection for the host, Communications Server IP filtering can provide a secondary line of defense.

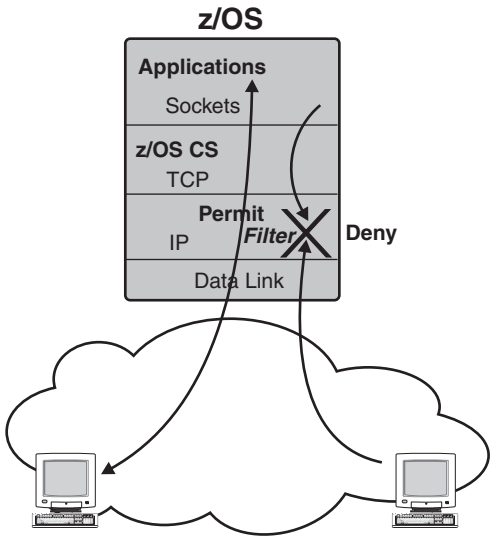


Figure 14. IP filtering at the z/OS communication endpoint

Protecting data in the network

Network security principals

Cryptography: The foundation of good security

The foundation of good security methods begins with cryptography. Cryptography keeps your data and communications secure using techniques such as encryption, authentication, and data integrity. Encryption services protect sensitive data from being read by other than the intended receiver. Cryptographic authentication and data integrity services allow communicating hosts to detect if data is altered in transit. Public key cryptography can identify and authenticate hosts or users. Public key cryptography can also be used in the secure creation of symmetric session keys for both security endpoints. Once a secure session is created, successful data authentication and decryption occur only if both hosts have the correct session keys.

End to end security

Cryptographic security solutions can be applied to a portion of the data path or end to end, whichever is appropriate for your security policy. Generally, the greatest degree of security is provided when cryptographic methods are used end to end. However, if only portions of the data path are considered untrusted by an enterprise (such as the Internet) it may be adequate to protect only the untrusted portion with cryptography. z/OS offers security protocols that can be configured to protect portions of the data path or the entire data path.

Workload-based security deployment

In making a security protocol selection, an important consideration is the application workload to be protected. In order to illustrate this concept, it is helpful to understand where various protocols are implemented from a protocol layering perspective.

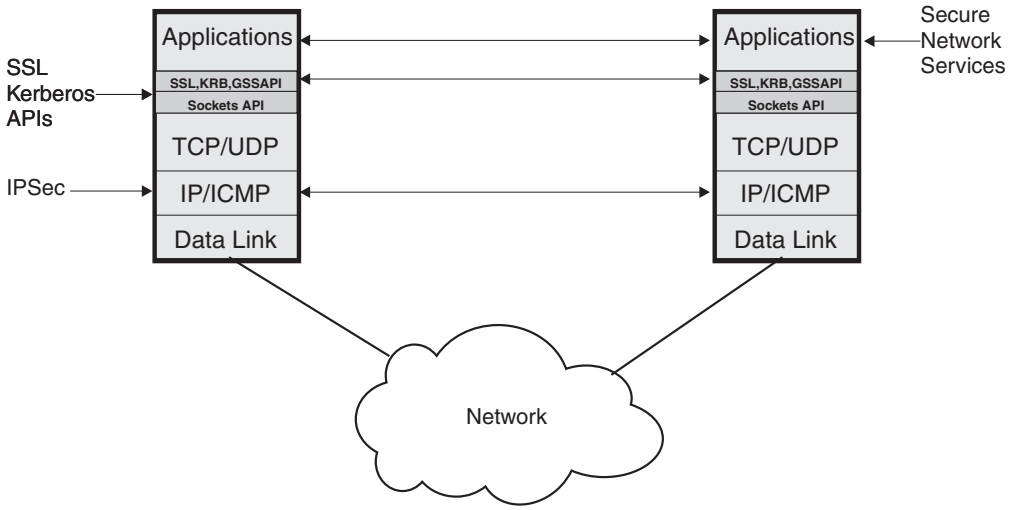


Figure 15. Security protocols from a protocol layering perspective

Existing workload: The network layer is the lowest layer in the protocol stack where end to end security over multiple hops can be applied. Network layer security protocols provide blanket protection for upper-layer application data without requiring modification to the application. IPSec is implemented at the network layer and provides authentication, integrity, and data privacy between any two IP entities. IPSec can protect a segment of the data path (e.g., between two routers), or it can secure the data path end to end. Because IPSec is applied at the IP layer, it is a connectionless security protocol and is applied on a per packet basis.

Secure Sockets Layer (SSL) is another popular security protocol implemented above the transport layer at the application interface layer. TCP applications must be modified to use SSL. SSL requires a reliable transport layer and is therefore not used for UDP applications. SSL provides authentication, integrity, and data privacy. SSL, originally used to secure traffic between a Web browser and Web server, can also secure other applications. SSL is a connection-oriented security protocol and protects all data on a connection or session.

The Communications Server has an SSL-enabled TN3270 server, thus allowing secure access to existing SNA applications being accessed over an IP network. Serving as a protocol gateway between the IP network and the SNA network, the SSL-enabled TN3270 server protects the data path in the IP network from the TN3270 client all the way to the z/OS TN3270 server. If the TN3270 Server resides on a different host from the target SNA application, SNA Session Level Encryption can be used to secure the SNA portion of the data path. SNA application data can be protected without modification to the SNA applications.

New workload: For new applications, security can be built-in. One method of building security into the application on z/OS is to use z/OS System SSL and Kerberos.

Newer versions of network services such as SNMPv3 and Secure DNS, which are supported by the Communications Server, have security built into the application protocol using standards-based specifications for secure interoperability.

Network security protocols

IPSec and VPNs

For more information about IPSec and VPNs, refer to *z/OS Security Server Firewall Technologies*.

The IPSec solution: IPSec is defined by the IPSec Working Group of the IETF. It provides authentication, integrity, and data privacy between any two IP entities. Management of cryptographic keys and security associations can be either manual or automated via an IETF defined key management protocol called Internet Key Exchange (IKE).

IPSec uses IP filtering to determine which traffic should be protected by IPSec. A type of permit rule specifies *permit with IPSec*. The IP filters represent IP security policy to the stack by specifying the traffic that requires IPSec. The filters are also used in locating the outbound IPSec security association, and for verifying that inbound traffic was received using the correct security association.

IP filtering can be used to avoid the overhead of multiple security protocols when alternate security protocols are used to secure specific applications. For example, you might want to exclude Web traffic (based on the well-known secure port of the Web server - port 443) from IPSec coverage because you would like to use SSL.

IPSec provides the flexible building blocks that can support a variety of configurations. Because an IPSec security association can exist between any two IP entities, it can protect a segment of the path or the entire path.

IPSec allows the creation of Virtual Private Networks (VPN). A VPN enables an enterprise to extend its network across a public network such as the Internet through a secure tunnel (or security association). IPSec and VPN enable the secure transfer of data over the public Internet for same-business and business-to-business communications, and protect sensitive data within the enterprise's internal network. The figure below shows some of the typical IPSec configurations. In this figure, IPSec security associations are shown between two firewalls, between client and firewall, and between client and zSeries server.

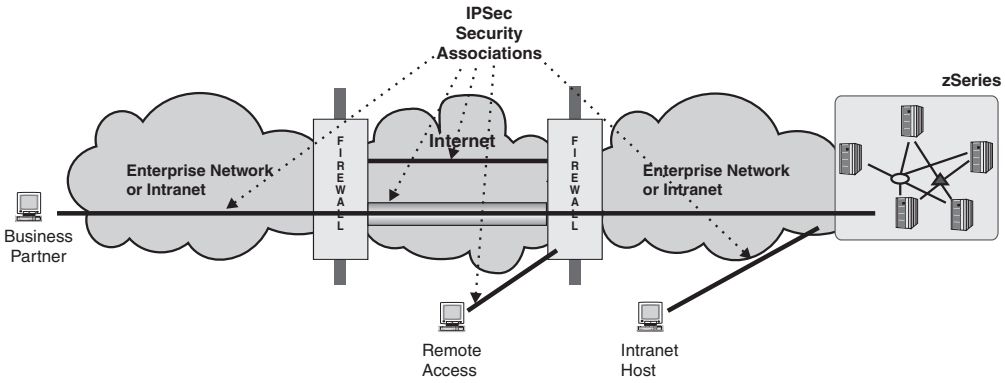


Figure 16. e-business scenarios with Virtual Private Networks

IPSec concepts and components:

Security Associations: The concept of a Security Association (SA) is fundamental to IPSec. An SA is a logical connection between any two IPSec systems. The SA defines the security services for traffic that it carries. The scope of protection of an

SA can vary. It can be *wide*, which means that the SA protects traffic for multiple connections (e.g., all traffic between two hosts). It can be *narrow*, which means that the SA protects traffic for a single connection.

An SA setup must occur before data is sent over a network. This setup can be accomplished by either configuring the SA manually, or creating the SA dynamically using the IKE protocols.

An SA can be in either of two modes:

- **Transport** mode is used by a host in cases where the data endpoint addresses are the same as the SA endpoint addresses. In this mode, the IPSec protocol header is inserted after the IP header and before the payload of the original IP datagram.
- **Tunnel** mode must be used whenever the SA endpoint addresses differ from the connection endpoint addresses. For this reason, SAs that start or end in firewalls that do not own the connection endpoint address are always tunnel mode. In this mode, the original IP datagram is made the payload of a newly constructed datagram.

IPSec has three major components:

- IP Authentication Header (AH)
- IP Encapsulating Security Protocol (ESP)
- Internet Key Exchange (IKE)

IP Authentication Header (AH): AH provides data integrity, data origin authentication, and an optional replay protection service. Data integrity is ensured by using a message digest generated by an algorithm such as HMAC-MD5 or HMAC-SHA. Data origin authentication is ensured by using a shared secret key to create the message digest. Replay protection is provided by using a sequence number field with the AH header. AH authenticates IP headers and their payloads with the exception of certain header fields that can be legitimately changed in transit such as the Time To Live (TTL) field. The following diagram shows the additional headers added as the result of AH processing and the scope of the authentication.

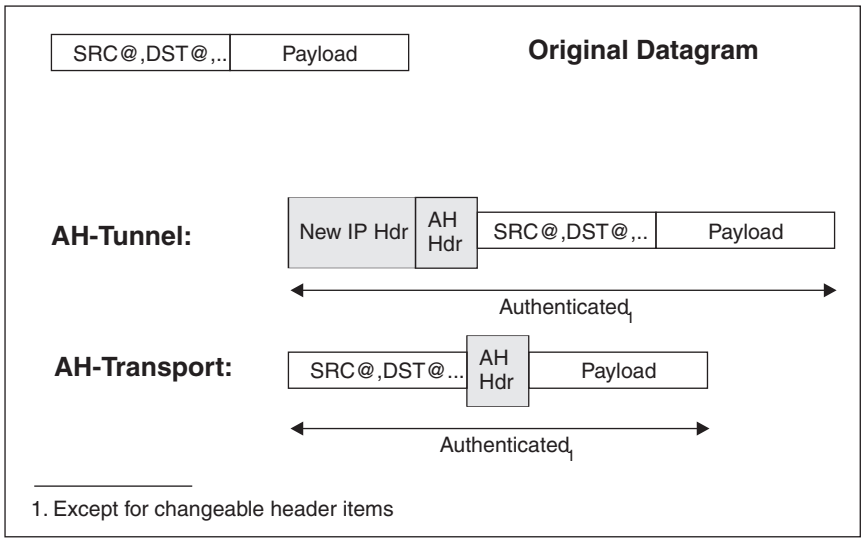


Figure 17. IPSec AH protocol header formats and security coverage

IP Encapsulating Security Protocol (ESP): ESP provides data confidentiality (encryption) and authentication (data integrity, data origin authentication, and replay protection). ESP can be used with confidentiality only, authentication only, or both confidentiality and authentication. When ESP provides authentication functions it uses the same algorithms as AH, but, the coverage is different. The following diagram shows the additional headers added as the result of ESP processing and the scope of the authentication and encryption.

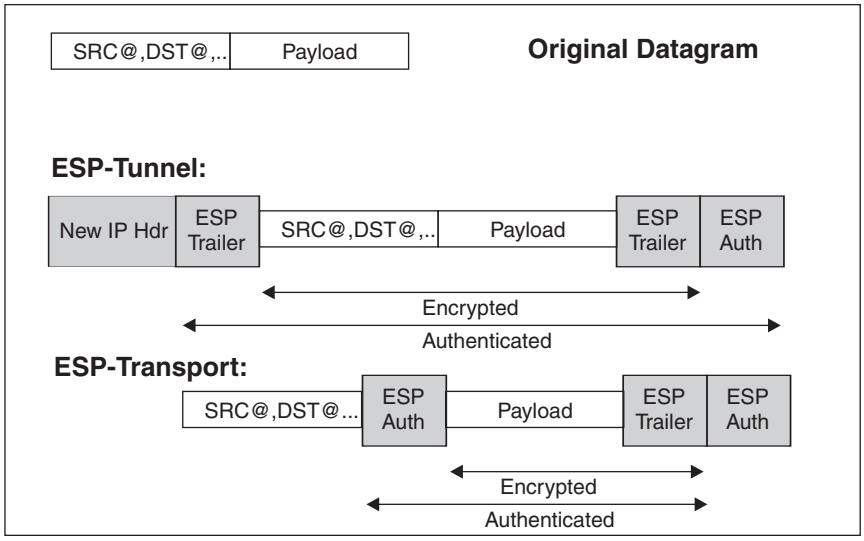


Figure 18. IPSec ESP protocol header formats and security coverage

Internet Key Exchange (IKE): IKE supports automated negotiation of SAs and automated generation and refreshing of cryptographic keys. The secure exchange of keys is the most critical factor in establishing a secure communications environment.

IKE operates at the application layer. It negotiates with its IKE peer to create two types of security associations called Phase 1 and Phase 2. IKE uses Phase 1 SAs to protect IKE flows. IPSec uses Phase 2 SAs to protect data transmissions. Once a Phase 2 SA is negotiated, IKE installs the Phase 2 SA into the stack so IPSec can use the SA to protect IP packets.

There are several methods by which IKE hosts can authenticate their IKE peers. Two of these methods are Pre-shared Key and RSA Signature. With Pre-shared Key, each IKE host is initially set up with a key that is used for authentication. RSA Signature uses a digital X.509 certificate for authentication. RSA Signature is a more scalable solution. Pre-shared Key requires that each host be keyed with every potential IKE partner key. With RSA Signature, each host is configured with its own host certificate and a certificate for the mutually trusted certificate authority that signed the host certificate.

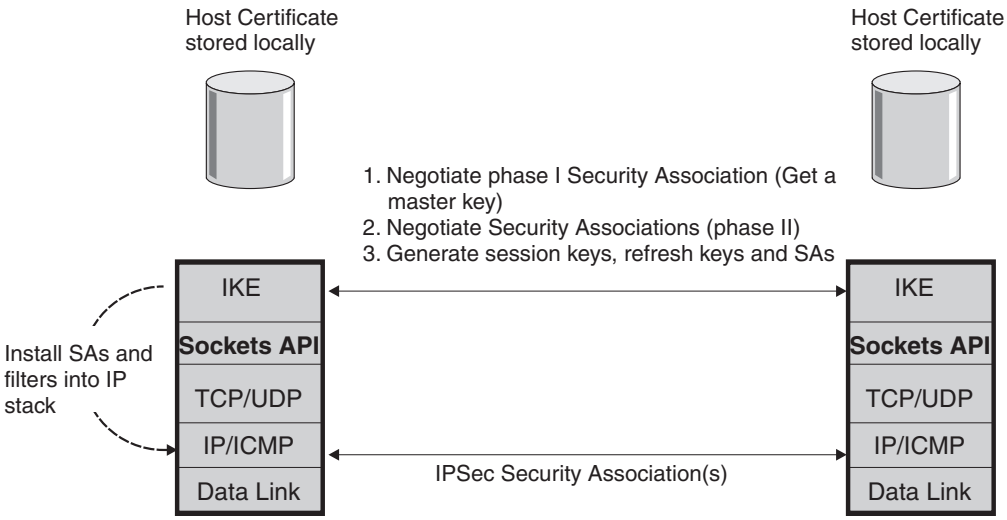


Figure 19. IPSec and IKE overview

Since the IKE protocols deal with initializing keys, they must be capable of running over links where no security can be assumed to exist. IKE addresses the problem of *secure key distribution* by automatically deriving the keying material using a Diffie-Hellman exchange during the Phase 1 IKE negotiation. This automatic creation and distribution of the key during Phase 1 eliminate the need to manually distribute the session key between remote sites. Besides the obvious administrative advantage of IKE, the manual method of key distribution is prone to key compromise.

In addition, IKE *non-disruptively refreshes the session keys* based on the security policy of the installation. IKE specifies that this can be based on time (lifetime) and/or bytes transmitted (lifesize). IKE provides a property called Perfect Forward Secrecy (PFS). If PFS is used, each Phase 2 key is derived independently through a separate Diffie-Hellman exchange. With PFS, if a single key is compromised, the integrity of subsequently generated keys is not affected. Manual IPSec has no key refresh capabilities unless the security associations are deactivated, reconfigured with the new key, and then reactivated. Because of the disruptive nature of key refresh with manual IPSec, key lifetimes are defined as much larger values thus, increasing the security exposure.

z/OS IPSec and VPN support: The Communications Server provides z/OS IPSec support. The Security Server provides Internet Key Exchange (IKE) support and Virtual Private Network (VPN) configuration. Together these z/OS elements combine to provide VPN support for z/OS.

z/OS provides support for the latest IETF RFCs (2401-2406, 2409, 2410) including Triple DES for strong encryption. A crypto coprocessor provides hardware assist for IPSec encryption and decryption. Both IPSec transport and tunnel modes are supported. IKE supports both pre-shared key and RSA Signature (which uses host-based X.509 certificates) methods of authentication. The z/OS IKE certificate is stored in RACF.

Configuration considerations for IPSec: In order to enable this support, you must specify the FIREWALL option on the IPCONFIG statement.

When you configure a mixture of secure and nonsecure adapters for z/OS CS and filter rules in your IPSec policy do not have an interface value of BOTH, you should ensure that all routes to the destinations in a single filter rule go through adapters with the same security level (for example, either secure or nonsecure).

For more information on using IPSec with Dynamic VIPAs, see “Sysplex Wide Security Associations” on page 228.

SSL and TLS

The SSL protocol provides data encryption, data origin authentication, and message integrity. It also provides server and client authentication using X.509 certificates. SSL begins with a handshake during which the server is authenticated to the client using X.509 certificates. Also, the client can optionally be authenticated to the server. During the handshake, security session parameters, such as cryptographic algorithms, are negotiated and session keys are created. After the handshake, the data is protected during transmission with data origin authentication and optional encryption using the session keys.

The cryptographic algorithms that are used for the SSL session are based on the algorithms the server and client are willing to use. During the SSL handshake, the client and server exchange a list of algorithms. The algorithm selected is based on the best match between the client’s list and the server’s list. The selectable algorithms can be limited by configuring a subset of allowable algorithms at the server. Servers can support encryption using Triple DES as well as other encryption algorithms (RC2, RC4, and DES). A hardware crypto coprocessor, if available, is used for DES and Triple DES encryption.

SSL requires a server X.509 certificate, which is stored in its certificate keyring. The certificate is used as part of the SSL handshake server authentication process. The client validates the server certificate. SSL optionally uses a client X.509 certificate that is used as part of the SSL handshake client authentication process. In order to use client authentication, the client must have a client X.509 certificate. Successful client authentication requires that the Certificate Authority (CA) that signed the client certificate be considered trusted by the server. To be considered trusted, the certificate of the CA must be in the keyring of the server.

Refer to “Transport layer security” on page 320 for detailed information on obtaining certificates.

SSL is not defined by the IETF. TLS is based on SSL and is defined by the IETF as RFC 2246.

TN3270 SSL: The Communications Server provides an SSL-enabled TN3270 server that protects the data path in the IP network to the z/OS TN3270 server using the SSL protocol. IBM Host On Demand and PCOMM provides a TN3270 client that is enabled for SSL.

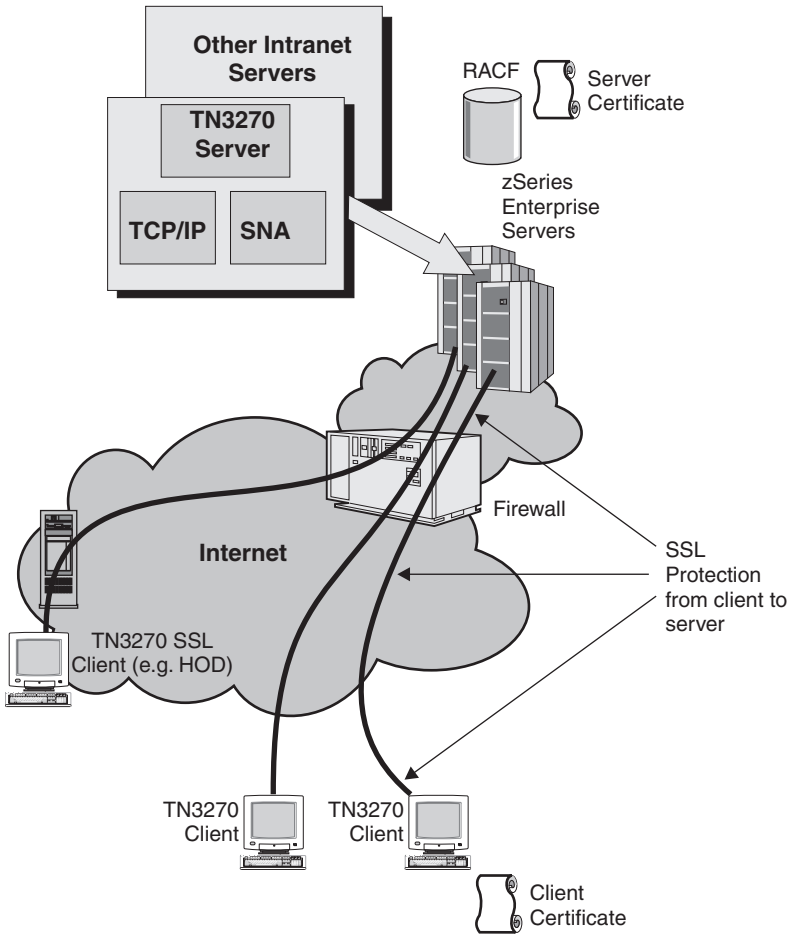


Figure 20. TN3270 SSL overview

The Communications Server TN3270 SSL support provides several extensions for RACF-based access control to the TN3270 server. These extensions prevent a client from seeing the USSMSG (log on screen) unless the client is authorized. In order to use this support, the client certificate must be defined to RACF using RACF digital certificate services. The first level of authorization checking verifies that the RACF userid represented by the client certificate is defined to RACF. The next level of authorization requires that this RACF userid be permitted to access the TN3270 server port. In this case, the TN3270 server port is represented as a RACF resource using the SERVAUTH class.

Multiple port support: One method of enabling a mix of SSL and non-SSL traffic is to use TN3270 multiple port support. Using the multiple port support, separate ports can be defined with one port being dedicated to non-SSL traffic and another port dedicated to SSL traffic. Ports designated as SECUREPORT are capable of using SSL. The following diagram illustrates the use of multiple ports. In this case, intranet clients are not required to use SSL. These clients connect to the BASIC port (port 23 in this example). All clients connecting from the Internet are required to use SSL. These clients use the SECUREPORT (port 1023 in this example). Packet filtering is used at the firewall that separates the intranet and the Internet to control access to the TN3270 ports. In order to prevent Internet access to the BASIC port, port 23 is blocked at the firewall. The SECUREPORT, port 1023, is permitted at the firewall. In this scenario, the best security is achieved when SSL client authentication with the TN3270 RACF extensions is used. This support ensures that the client has authority to attempt to log on to SNA applications through TN3270.

Regardless of the method of authentication used, the SNA application should identify and authenticate the end user using RACF before any application access is granted. SSL encryption services, if used, would encrypt the user ID and password.

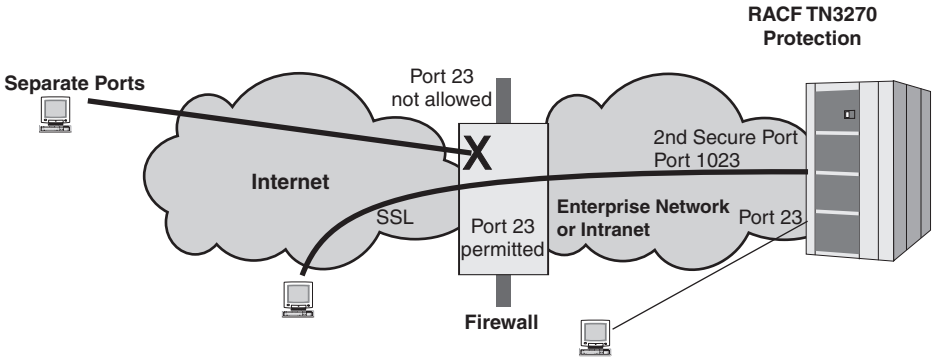


Figure 21. Using multiple TN3270 ports to separate SSL and non-SSL traffic

This next diagram illustrates how IPSec and SSL can be combined to provide a more secure remote access from the Internet to SNA applications than is depicted in the previous diagram. In this scenario, IPSec AH protocol is used between the user's PC and the firewall for authentication. The firewall is open for port 1023 for traffic that is authenticated with IPSec only. The firewall would discard traffic for port 1023 that cannot be authenticated by IPSec. The additional security provided by IPSec protects the zSeries from unauthorized access attempts and denial of service attacks by hosts outside the VPN.

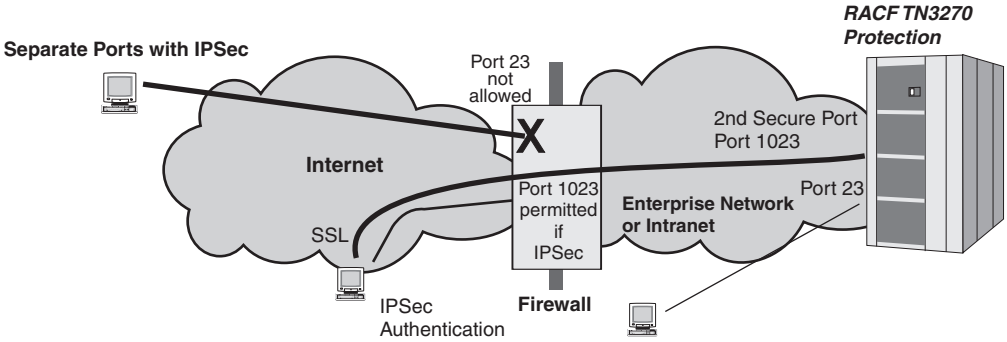


Figure 22. Combining TN3270 SSL with IPSec client-to-firewall authentication

TN3270 use of single port for SSL and non-SSL connections: A single port can be used to support a mix of SSL and non-SSL traffic. In this case the port is designated as SECUREPORT. In order to support the configuration of various SSL security policies for a single port, the SECUREPORT designation defines the port to be capable of using SSL, rather than the port must use SSL. The PARMSGROUP and BEGINVTAM blocks are used to specify the connection type (CONNTYPE) associated with a subset of the port's connections. A PARMSMAP statement is used to associate the PARMSGROUP information with specific IP address, hostname, or linkname. CONNTYPE specifies the SSL policy for the connections that are associated with it.

The TN3270 server supports both negotiated and non-negotiated SSL. TN3270 negotiated SSL is an IETF defined extension to the TN3270 protocol. With TN3270 negotiated SSL, the decision to use SSL for a connection is based on the outcome of a negotiation between the TN3270 client and server using TN3270 protocols.

This negotiation is performed after the TN3270 connection is established, and if SSL is negotiated, the SSL handshake is performed. With non-negotiated SSL, an SSL handshake is required immediately after connection establishment. Concurrent use of both TN3270 negotiated and non-negotiated SSL connections are allowed for a single port.

The following diagram illustrates the use of a single TN3270 port that allows a mix of SSL and non-SSL traffic. In this case, intranet clients are not required to use SSL. All clients connecting from the Internet are required to use SSL. Both intranet and Internet clients connect to the SECUREPORT (port 23 in this example). In this scenario, IPSec AH protocol is used between the user's PC and the firewall for authentication. The firewall is open for port 23 for traffic that is authenticated with IPSec only. The firewall would discard traffic for port 23 that IPSec cannot authenticate. In this scenario, packet filtering without IPSec cannot be used at the firewall that separates the intranet and the Internet to control access on the basis of port since only one port is used. Without IPSec AH, all access control checks are deferred to the TN3270 Server. The additional security provided by IPSec at the firewall protects the zSeries from unauthorized access attempts and denial of service attacks by hosts outside the VPN.

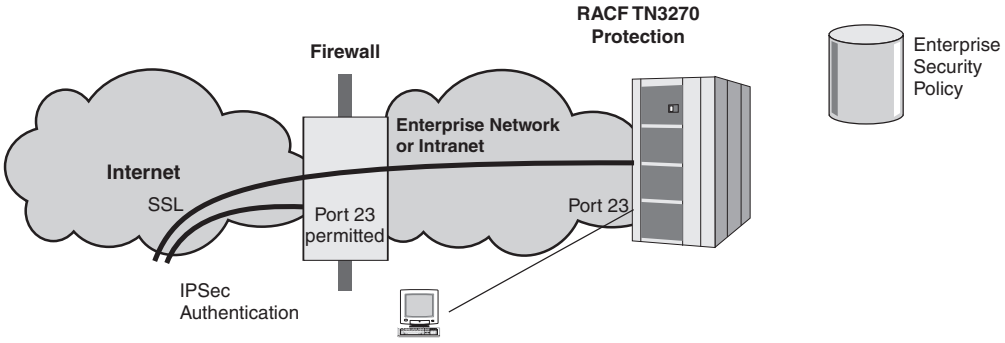


Figure 23. TN3270 SSL and non-SSL traffic using a single TN3270 port

Express Logon Feature (ELF): With emulator products, the traditional method of authenticating the user is through user ID and password which is kept in sync with the host access control facility (RACF, ACF/2, AS/400® user management, etc.). The Express Logon Feature simplifies user ID and password administration for users signing on to SNA applications using TN3270. ELF allows an end user to use an SSL-authenticated X.509 certificate for authentication to the SNA application instead of using a user ID and password. ELF requires IBM Host Integration software. The Host Integration requirements depend on the configuration.

There are two network designs available; a two-tier or a three-tier approach. Both are discussed in Appendix C, "Express Logon Feature (ELF)" on page 749.

TLS-enabled FTP: The Communications Server FTP server and client support Transport Layer Security (TLS). This support enables secure file transfer by providing data privacy, message authentication, and message integrity services for data sent and received using the FTP control and data connections.

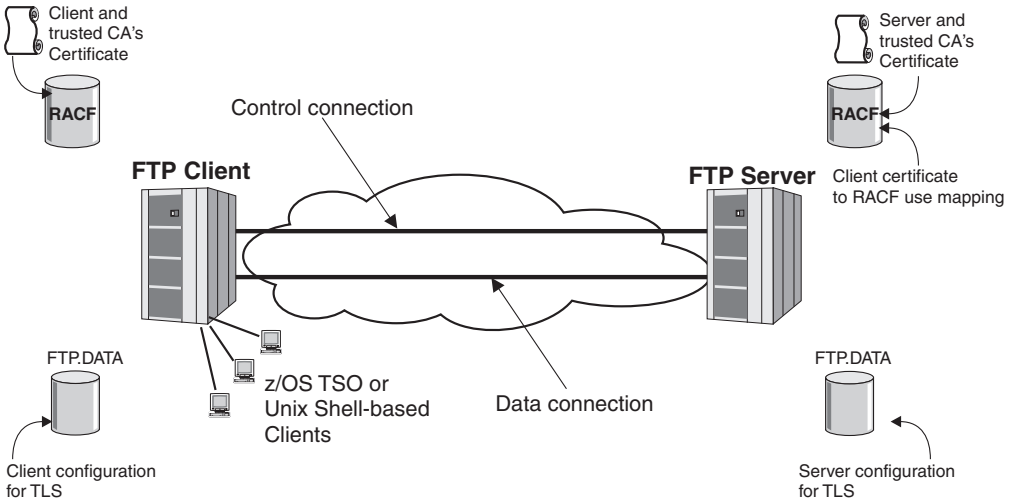


Figure 24. FTP client and server TLS overview

The TLS-enabled FTP server can be configured to run in two modes. Conditional mode allows an installation to use a single port for both TLS and non-TLS FTP control connections. In conditional mode, the FTP client and server negotiate the use of TLS based on a subset of the FTP security negotiation functions documented in RFC 2228. Once the use of TLS is negotiated, the TLS handshake is performed which establishes the TLS session and negotiates security parameters and session keys. Unconditional mode allows an installation to use a separate port for all TLS traffic. Port 990 is the port designated for control connections for unconditional TLS mode. With unconditional mode, it is assumed that TLS is required, and after the FTP control connection is made, the TLS handshake is performed.

TLS secures the control connection and optionally the data connection. TLS for the data connection requires a TLS session for the control connection. FTP server configuration controls whether the FTP server requires TLS for the control and data connections. This TLS protection by connection type is negotiated during the FTP RFC 2228 negotiation that precedes the TLS handshake. During the lifetime of the control connection, the use of TLS or no TLS for the data connection can be requested by the FTP client using the FTP RFC 2228 commands.

FTP TLS optionally authenticates the client during the TLS handshake using a client X.509 certificate. FTP server configuration specifies whether TLS client authentication is required and what type of validation of the certificate is required. For example, the FTP server can be configured to map the client certificate to a RACF userid and then verify that the userid associated with the certificate matches the userid entered by the end user.

Configuration to control TLS capabilities and options for both FTP client and server TLS are stored in the FTP.DATA data set.

Kerberos

Kerberos is a network authentication protocol that is designed to provide strong authentication for client/server applications using secret-key cryptography. The Kerberos network authentication protocol assumes that services and workstations communicate over an insecure network. It allows clients and servers to do either one way, or two way (mutual) authentication. It allows for data encryption and prevents passwords from having to be retyped to access networked services and

also prevents their transmission in plain text over the network. This feature can help reduce the need to manage multiple passwords.

z/OS CS no longer ships Kerberos V4. z/OS Security Server ships a different Kerberos, Version 5. Because Security Server Kerberos does not require DCE login and eliminates the need for multiple registries, it is recommended that new applications be written to Kerberos Version 5 and use z/OS Security Server.

The following Communication Server IP applications now include support for Kerberos Version 5 security protocol:

- The UNIX System Services Telnet Server now allows clients supporting Kerberos Version 5 (as described in RFC 1416) to log in to the shell environment using Kerberos as the authentication protocol.
- The FTP client and Server now support connections to or from other clients and servers supporting Kerberos Version 5 authentication for the FTP protocol (as described in RFC 2228).
- The UNIX System Services RSH server can now also be configured to support client authentication using Kerberos from RSH clients supporting Kerberos Version 5.

OSPF authentication

Communications Server OSPF (Open Shortest Path First) dynamic routing protocol supports message authentication and message integrity of OSPF routing messages through the use of the OSPF MD5 Authentication security protocol as defined by RFC 2328. OSPF MD5 Authentication ensures that an unauthorized IP resource cannot inject OSPF routing messages into the network without detection, thus ensuring the integrity of the routing tables in the OSPF routing network.

OMPROUTE computes a secure MAC for the routing message using the MD5 algorithm. This MAC is sent with the routing message so that the message can be authenticated by the receiver.

Secure DNS

The Communications Server supports DNS at the Version 9.1 of BIND. This level of DNS has built-in security features, DNSSEC and TSIG.

DNSSEC: DNSSEC ensures that DNS query results are not spoofed and in fact originate from a trusted DNS. DNSSEC defines extensions to DNS that provide data integrity and authentication to security aware resolvers and applications through the use of cryptographic digital signatures. DNSSEC is defined by the IETF in RFC 2535.

TSIG: TSIG is a protocol for Secret Key Transaction Signatures for DNS. This protocol allows for transaction level authentication using shared secrets and one way hashing. It authenticates dynamic updates as coming from an approved client, or responses as coming from an approved recursive name server.

SNMPv3

z/OS CS SNMP supports SNMPv3. The legacy community-based protocols SNMPv1 and SNMPv2 are also supported. SNMPv3, defined in RFCs 2570 through 2575 is the standards-based solution for SNMP security. It is categorized as a User-based Security Model (USM) which provides different levels of security based on the user accessing the managed information. To support this security level, the SNMPv3 framework defines several security functions, such as USM for authentication and privacy, and view-based access control model (VACM) which provides the ability to limit access to different MIB objects on a per-user basis, and

the use of authentication and data encryption for privacy. However, SNMP is not just enhanced security. It defines an architecture for SNMP management frameworks, with the intent that pieces of the architecture can advance over time without requiring the entire structure to be rewritten. For that reason, three major subsystems are defined:

- Message processing subsystem
- Security subsystem
- Access control subsystem

The framework is structured so that multiple models can be supported concurrently and replaced over time. For example, although there is a new message format for SNMPv3, messages created with the SNMPv1 and SNMPv2 formats can still be supported. Similarly, the user-based security model can be supported concurrently with the community-based security models previously used.

Security Event Reporting

Integrated Intrusion Detection Services (IDS)

Intrusion is a broad term encompassing many undesirable activities. The objective of an intrusion may be to acquire information that a person is not authorized to have (*information theft*). It may be to cause business harm by rendering a network, system or application unusable (*denial of service*). Or it may be to gain unauthorized use of a system as a stepping stone for further intrusions elsewhere. Most intrusions follow a pattern of information gathering, attempted access and then destructive attacks. Some attacks can be detected and neutralized by the target system. Other attacks cannot be effectively neutralized by the target system. Many of the attacks also make use of spoofed packets which are not easily traceable to their true origin. Many attacks now make use of unwitting accomplices - machines or networks that are used without authorization to hide the identity of the attacker. For these reasons, detecting information gathering, access attempts and attack accomplice behaviors is a vital part of intrusion detection.

Attacks can be initiated from outside the internal network or from inside the internal network. Particularly vulnerable is an open system such as a public Web server or any machine that is placed in service to serve those outside the internal network. A firewall can provide some level of protection against attacks from outside. However, it cannot prevent attacks once the firewall has authorized an external host to communicate with hosts in the internal network, nor can it provide protection in the case where the attack is initiated from inside the network. In addition, end to end encryption limits the types of attacks that can be detected by an intermediate device such as a firewall.

An Intrusion Detection System can provide detection of some types of attacks. Common intrusion detection system types currently deployed are network sniffers or sensors and vulnerability scanners. Sniffers, placed at strategic points in the network (in front or behind a firewall, in the network, or in front of a host), operate in promiscuous mode, examining traffic real-time that passes through on the local network. Sniffers use *pattern matching* to try to match a packet against a known attack which is expressed as an *attack signature*. Sniffers work best against single packet attacks. Limitations are that they cannot deflect the attacking packet, and they cannot evaluate against encrypted data. Scanners do not detect intrusions in real-time. They examine a system periodically looking for vulnerabilities or evidence of intrusion. Some scanners evaluate historical data and can identify behavioral anomalies and patterns associated with intrusions.

The z/OS Communications Server provides Intrusion Detection Services (IDS) which enable the detection of attacks and the application of defensive mechanisms on the z/OS server. The focus of IDS is self-protection. IDS can be used alone or in combination with an external network-based Intrusion Detection System. The IDS is integrated into the z/OS Communications Server stack and can provide the following functions unavailable from an external Intrusion Detection System.

- z/OS CS IDS evaluates data that has been encrypted by IPSec end to end after decryption on the target server system.
- z/OS CS IDS avoids the overhead of per packet examination against a table of signatures for many known attacks. This is accomplished by integrating the attack detection probes into existing error detection logic. This detection is done in real-time. IDS policy is examined when an attack is detected to determine the action to be taken.
- z/OS CS IDS detects statistical anomalies real-time. Real-time detection is achieved since it is easier for the target system to keep stateful data/internal thresholds and counters.
- z/OS CS IDS implements prevention type of policies that are executed on the system that is the target of the attack. Prevention policies include packet discard and connection limiting.

The IDS is policy driven and the policies are kept in LDAP. These policies determine what actions to take for various IDS events. IDS events detected include scans, single packet attacks against the TCP/IP stack, and flooding. Actions include packet discard, connection limiting, and reporting. IDS events can be recorded in syslog files and/or the console. IDS statistics can be recorded in syslog. Packet traces can be taken to document suspicious activities. The TRMDSTAT command provides summary and detailed reporting of IDS events and statistics.

The following figure shows the z/OS CS IDS architecture.

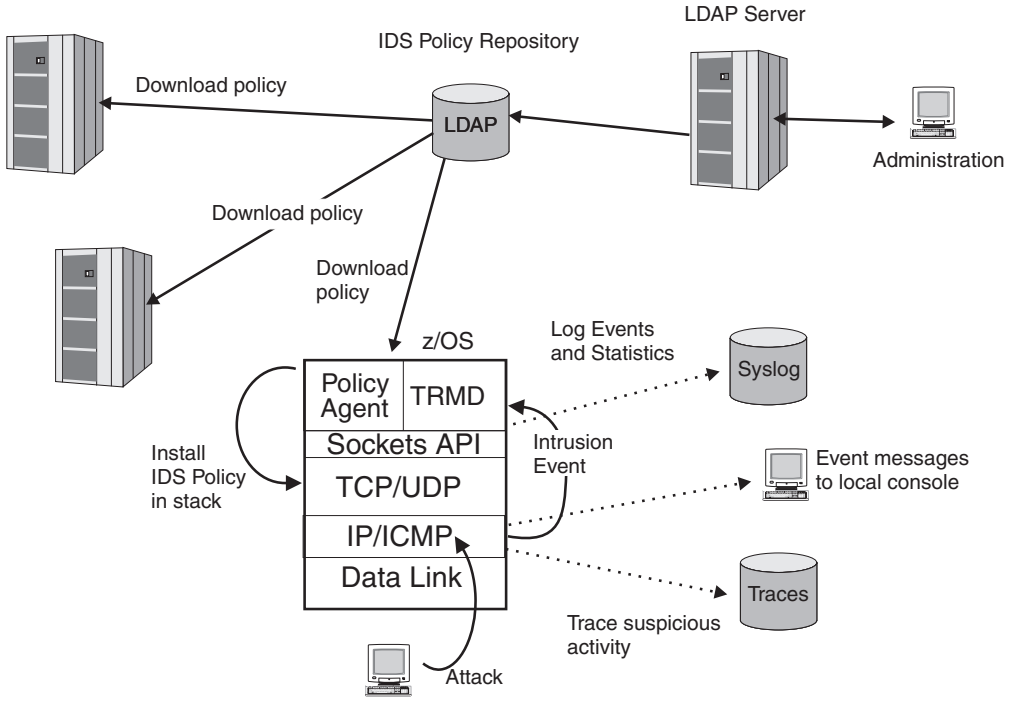


Figure 25. Intrusion Detection Services overview

Chapter 3. Customization

Before you begin customizing, it is assumed that you know what configuration data sets are used by the TCP/IP address space, their search order, and considerations for what type of TCP/IP stack you will be running in your environment (for example, Enterprise Extender (EE) and multiple stacks). See Chapter 1, “Configuration overview” on page 3 for this information.

After reading this chapter, you will know how to configure and start syslogd and the TCP/IP stack. You should understand the relationships of TCP/IP configuration files as they apply to the TCP/IP address space. The four main configuration files that you will be working with are:

- TCPIP.DATA
- PROFILE.TCPIP
- HOSTS.LOCAL
- ETC.IPNODES

You should be able to use the following commands to verify customization:

TSO PING, z/OS UNIX oping, and z/OS UNIX ping
: Sends IP datagrams to a specified destination host, requesting a reply, and measures the round trip time. This helps you to verify the interfaces defined to the TCP/IP address space.

TSO NETSTAT, z/OS UNIX onetstat, and z/OS UNIX netstat
: Queries TCP/IP about the network status of the local host. With NETSTAT, you can verify most TCP/IP customization values that can be set from the PROFILE.TCPIP.

TSO HOMETEST
: Verifies your host name and address configuration.

TSO TRACERTE, z/OS UNIX otracert, and z/OS UNIX traceroute
: Displays the route that a packet takes to reach a requested destination.

Configuring the syslog daemon (syslogd)

Configuration statements

The syslogd processing is controlled by a configuration file called /etc/syslog.conf (see the following sample file) in which you define logging rules and output destinations for error messages, authorization violation messages, and trace data. Logging rules are defined using a *facility* name, a *priority* code, and the user ID and job name of the program that generated the message. The *facility* name and *priority* code are passed on the logging request from an application when it wants to log a message. The user ID and job name are provided by the system. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about logging rules.

As shown in the following sample /etc/syslog.conf file, comments can be added to the configuration file by placing the # character in column one of the comment line. Everything following the # character is treated as a comment. This sample is available in /usr/lpp/tcpip/samples/syslog.conf in the HFS.

```
#  Licensed Materials - Property of IBM
#  5694-A01
#  (C) Copyright IBM Corp. 1992,  2002
```

```
#  Status = CSV1R4
#
# /etc/syslog.conf - control output of syslogd
#
# The # sign begins a comment which extends to the end of the line.
#
# Blank lines are ignored.
#
# Rules in this file specify types of messages which syslogd will
# store, and where syslogd will store it.
#
# See IP Configuration Reference for detailed information about
# the syntax.  These comments are meant to provide only a general
# overview.
#
# Four criteria can be used to select messages for processing:
#
# 1) user ID associated with application generating the message
#
#    * can be specified for the user ID if the user ID is not
#    important.
#
# 2) job name of application generating the message
#
#    * can be specified for the job name if the job name is not
#    important.
#
# 3) facility of the message, as specified by the application
#
#    This is user, mail, news, uucp, daemon, auth, cron, lpr, or
#    local0-local7.  Consult the documentation for the application
#    to determine which facility the application specifies.
#
#    A special facility, mark, specifies that syslogd should log
#    mark messages on a regular basis.  These can be used to verify
#    that syslogd was operational during a specific time interval.
#
# 4) priority of the message, as specified by the application
#
#    This is emerg, panic, alert, crit, err, error, warn, warning,
#    notice, info, or debug.
#
#    A special priority, none, specifies that messages with the
#    specified user ID, job name, or facility should not be
#    selected.
#
# These criteria are specified together as
#
#    userid.jobname.facility.priority
#
# or, if user ID and job name are both *, as
#
#    facility.priority
#
# This can be combined in a series as
#
#   userid.jobname.facility.priority;userid.jobname.facility.priority
#
# The criteria for selecting messages for processing are combined
# with a destination, which tells syslogd what to do with selected
# messages.
#
#   criteria    destination
#
# The destination can be a file, one or more user IDs, SMF, syslogd
# at a remote host, or all logged-in users.
#
```

```
# The following example stores messages with facility daemon or
# local1 in the file /directory/logfile.
#
#   daemon.*;local1.*   /directory/logfile
#
# The directory structure used in this sample configuration is
# expected to be created automatically by syslogd, with a new
# directory of log files for each day.  This requires two types
# of configurations outside of the scope of this configuration
# file:
#
# 1) syslogd command-line option
#
#    The syslogd -c command-line option should be enabled, causing
#    syslogd to create log files and directories if they do not
#    already exist.
#
# 2) cron job
#
#    A cron job should be utilized to wake up syslogd at the
#    beginning of each day to switch to new log files in a new
#    directory.  Here is the cron job definition:
#
#      1 0 * * * kill -HUP `cat /etc/syslog.pid`
#
#    This job should be defined for a user ID with UID zero so that
#    it has permissions to send the signal to syslogd.
#
#    See UNIX System Services Planning and UNIX System Services
#    Command Reference for more information about cron.
#
# A sample shell script is provided for removing log files which are
# a specifed number of days old.  It assumes the same directory
# structure which is used in this sample configuration.
#
# All example rules except for the last one are commented-out.  Some
# or all of the example rules will need to be changed for your
# environment.  Each example rule contains an explanation of changes
# which may be required.
#
######################################################################
#
# Write all messages with priority crit or higher to the MVS operator
# console.  See the UNIX System Services Planning manual for more
# information about the /dev/console special file.
#
# *.crit             /dev/console
#
######################################################################
#
# Write all messages from syslogd itself to the file
# /var/log/YYYY/MM/DD/syslogd.log and to the system console.
#
# Notes:
#
# a) If syslogd is invoked as a started task with job name
#    SYSLOGD, the name of the long-running syslogd job is
#    SYSLOGD1.  If syslogd is invoked from a shell script
#    (e.g., /etc/rc) with job name SYSLOGD, the name of the
#    long-running syslogd job is SYSLOGD followed by a
#    digit.
#
#    If syslogd runs with a different job name on your system, the
#    rule will have to be changed accordingly.
#
# b) During initialization, syslogd writes messages to
#    /dev/console.  These rules cover messages during steady-
```

```
#    state.
#
# *.SYSLOGD*.*.*      /var/log/%Y/%m/%d/syslogd
# *.SYSLOGD*.*.*      /dev/console
#
######################################################################
#
# Write all messages from inetd to the log file inetd and to the
# console.
#
# Notes:
#
# a) If inetd is invoked as a started task with job name INETD, the
#    name of the long-running inetd job is INETD1.  If inetd is
#    invoked from a shell script (e.g., /etc/rc) with job name INETD,
#    the name of the long-running inetd job is INETD followed by a
#    digit.
#
#    If inetd runs with a different job name on your system, the rule
#    will have to be changed accordingly.
#
# *.INETD*.*.*       /var/log/%Y/%m/%d/inetd
# *.INETD*.*.*       /dev/console
#
######################################################################
#
# Write all messages with priority err or higher from applications
# which specify facility "daemon" to the log file daemon.
# Because we chose to log messages from syslogd and inetd separately,
# we'll filter out those messages from this rule using special
# priority none.
#
# Notes:
#
# a) In this example, SYSLOGD followed by some other character is the
#    job name of syslogd.  If it is different on your system, change
#    the rule.
# b) In this example, INETD followed by some other character is the
#    job name of inetd.  If it is different on your system, change the
#    rule.
#
# daemon.err;*.SYSLOGD*.*.none;*.INETD*.*.none /var/log/%Y/%m/%d/daemon
#
######################################################################
#
# Write all messages from applications which specify facility "auth"
# to the log file auth.
#
# auth.* /var/log/%Y/%m/%d/auth
#
######################################################################
#
# Write all messages from applications which specify facility "mail"
# to the log file mail.
#
# mail.* /var/log/%Y/%m/%d/mail
#
######################################################################
#
# Write all messages with priority err and higher from otelnetd and
# other applications which specify facility "local1" to the log file
# local1.
#
# local1.err       /var/log/%Y/%m/%d/local1
#
######################################################################
#
```

```
# Write all messages from otelnetd and other applications which
# specify facility "local1" when running as user SMITH to the log file
# local1.smith.  This could be useful if, for example, otelnetd traces
# need to be collected for a problem which user SMITH is experiencing
# and you do not wish to collect otelnetd traces from all user IDs.
#
# SmITh.*.local1.*  /var/log/%Y/%m/%d/local1.smith
#
#####################################################################
#
# Write all messages with priority err and higher to SMF.  These will
# be stored in SMF record type 109.  SMF must be active and
# configured to accept record type 109.  The user ID associated with
# syslogd must have read access to BPX.SMF.  See UNIX System Services
# Planning for more information about BPX.SMF.
#
# *.err        $SMF
#
#####################################################################
#
# Write all messages with priority crit and higher to the syslogd on
# host 192.168.1.9. The host may be specified by IPv4 address or by
# a name that resolves to an IPv4 address.
#
# *.crit      @192.168.1.9
#
#####################################################################
#
# Write all messages with priority err and higher to log file errors.
#
# THIS EXAMPLE STATEMENT IS UNCOMMENTED.
#
*.err            /var/log/%Y/%m/%d/errors
#
```

Starting and stopping syslogd

Following is the syntax for the syslogd command:

syslogd [–f *conffile***] [–i][–u[–c[–d][–m** *markinterval***] [–p** *logpath***]**

syslogd recognizes the following options:

-f Configuration file name.

-i Do not receive messages from the IP network.

-u For records received over the AF_UNIX socket (most messages generated on the local system), include the user ID and job name in the record. In this case, a forward slash, the user ID, and the job name will follow the local host name for messages received over the AF_UNIX socket. The forward slash, which immediately follows the local host name, can be used to determine whether or not the user ID and job name is being recorded. If not recorded, a blank immediately follows the local host name. When user ID or job name is not available, *N/A* will be written in the corresponding field.

-c Create log files and directories automatically.

-d Run syslogd in debugging mode (see “Diagnosing syslogd configuration problems” on page 108 for more information).

-m Number of minutes between mark messages. The default value is 20 minutes. The following rule must be coded for each logfile that you want a mark record recorded in: mark.info.

-p Path name of z/OS UNIX character device for the datagram socket. The default value is /dev/log.

Note: This option is not used frequently. If you selected the -p option, syslogd will not function properly.

To specify the job name and pass the appropriate environment variables to the syslogd process, start syslogd using a shell script such as the following:

```
#
# Start the syslog daemon
#

export _BPX_JOBNAME='syslogd'
/usr/sbin/syslogd -f /etc/syslog.conf &
```

You can execute this shell script directly from the /etc/rc file to start syslogd at z/OS UNIX initialization.

If an incorrect argument or number of arguments is entered, syslogd exits and the return code is 1. In all other situations in which syslogd exits, the return code is 0.

To terminate syslogd, send a SIGTERM signal.

```
kill -s SIGTERM <PID>
```

To force syslogd to reread its configuration file and activate any modified parameters without stopping, send a SIGHUP signal. syslogd will continue to append log messages to the files you specify in /etc/syslog.conf.

```
kill -s SIGHUP <PID>
```

The syslog daemon stores its process ID in the /etc/syslog.pid file so that it may be used to terminate or reconfigure the daemon. For syslogd to successfully create this file, you must define the syslogd user ID as UID=0.

Note: If the BPX.SMF facility is defined and SMF records are to be written by syslogd, the user ID with which syslogd runs must also be permitted to SAF resource BPX.SMF. See SEZAINST(EZARACF) for more information.

Messages are read from the UNIX domain datagram socket and, unless the -i command line option is specified, the IPv4 Internet domain (AF_INET) datagram socket. AF_INET6 sockets are not supported. Kernel messages are not logged by syslogd in z/OS UNIX.

Note: For more information about the facilities used by z/OS CS functions, see Table 4 on page 40.

Offloading log files

z/OS CS includes a syslogd configuration file in /usr/lpp/tcpip/samples/syslog.conf, a REXX program for removing old log files in /usr/lpp/tcpip/samples/rmoldlogs, and a JCL procedure for starting syslogd in SEZAINST(syslogd). These are intended to be used together, though each may need to be customized for your installation.

The sample syslogd configuration file is installed in /usr/lpp/tcpip/samples/syslog.conf. It can be copied to /etc/syslog.conf after customization. If it is copied somewhere else, the syslogd -f command-line option must be used to tell syslogd where to find the configuration file.

The sample REXX program for removing old log files is installed in /usr/lpp/tcpip/samples/ezaslrol. It can be copied to an installation-defined directory after customization. The sample JCL procedure can be copied to an installation-defined library after customization.

The sample configuration uses date stamps in the names of directories of log files to organize log files by year (%Y), month (%m), and day (%d) as follows:

```
*.err      /var/log/%Y/%m/%d/errors
```

Log files for February 14, 2001, for example, would be stored in directory /var/log/2001/02/14. Variable substitution occurs using the LE C function strftime(). Variables are case sensitive. For more information and a complete list of variables, refer to *z/OS C/C++ Run-Time Library Reference*.

A cron job should be used to send the SIGHUP signal to syslogd every day at midnight so that it switches to a new set of files. The cron job should be created for a user ID with UID 0. The definition of the cron job is:

```
0 0 * * * kill -HUP `cat /etc/syslog.pid`
```

The log file names vary based on the day, so sending SIGHUP to syslogd after the day changes causes syslogd to create new files.

Because some messages sent just after midnight may be logged by syslogd before it processes the SIGHUP signal, it is possible that a few messages sent after midnight will be stored in the log files for the previous day.

The sample REXX program can be run daily to remove all log files older than the number of days specified in the program. Comments in the REXX program describe how to configure the number of days. The definition of a cron job to run the REXX program every day at 1:00 A.M. is:

```
0 1 * * * localdir/ezaslrol
```

localdir is the name of the installation-defined directory where the customized version of /usr/lpp/tcpip/sample/ezaslrol was copied.

Using syslogd for z/OS UNIX application programs

You can use the logging facilities of the syslogd server with your z/OS UNIX application programs. Include the syslog.h header file with C programs so that they can open a log facility, send log messages to syslogd, and close the facility:

```
#include <syslog.h>

1 openlog("oec", LOG_PID, LOG_LOCAL0);
2 syslog(LOG_INFO, "Hello from oec");
3 closelog();
```

1 Open a log facility with the name of local0. Prefix each line in the log file with the program name (oec) and the process ID.

2 Log an info priority message with the specified content.

3 Close the log facility name.

The preceding statements created the following line in the log file:

```
May 26 11:27:51 mvs18oe oec[3014660]: Hello from oec
```

For more information about the syslog function, refer to *Advanced Programming in the UNIX Environment*, published by Addison-Wesley or *z/OS C/C++ Run-Time Library Reference*.

Usage notes

- syslogd can be started only by a task or user with superuser authority.
- syslogd can be terminated using the SIGTERM signal.
- If you want syslogd to receive log data from remote syslogd servers, ensure that syslogd can bind to UDP port 514 by reserving that port for the syslogd job in your PROFILE.TCPIP data set. Ensure that the syslog service is defined in your services file or data set (for example, /etc/services). The following example port reservation in PROFILE.TCPIP assumes that syslogd runs as job syslogd1:

  ```
  PORT
  ...
  514 UDP syslogd1     ;syslogd daemon
  ...
  ```

  The following example shows the services file or data set file entry:

  ```
  syslog     514/udp
  ```
- If there is no TCP/IP transport active when syslogd starts or if TCP/IP is recycled, syslogd will establish or reestablish communication with TCP/IP when it becomes available.
- Configuration file errors are written to the operator console because initialization is not complete until the entire configuration file has been read.
- Facility mark is not affected by the *.priority usage. Mark messages are written only to the destinations of rules that specify mark.info.
- If a mark interval of zero minutes is specified, mark messages will be written every thirty seconds.

Diagnosing syslogd configuration problems

syslogd supports a debug mode, which is selected using the -d command-line option. In this debug mode, syslogd does not run as a daemon, but instead runs in the foreground and writes a large number of trace messages to STDOUT. These messages can be used to diagnose problems in the syslogd configuration or to collect documentation when reporting a syslogd problem to IBM support.

Note: Do not use the -d option for normal operations.

If you are running syslogd in batch with -d, debug output is written to SYSPRINT, SYSTERM, or SYSERR, whichever is found first. The sample syslogd procedure SEZAINST(syslogd) defines SYSPRINT so that debug messages are stored in the job output.

Use caution using -d when syslogd is started from /etc/rc. If -d is used in this way, the shell *and* operator must be used to run syslogd in the background. Otherwise, /etc/rc does not end and UNIX System Services initialization does not complete.

Also, use caution when using -d along with a port reservation statement for the syslogd port (UDP port 514) in the TCP/IP profile. The job name of syslogd might differ based on whether or not the -d option was specified. If a port reservation statement is coded based on the job name that syslogd uses without the -d option, syslogd might not be able to bind to the port when run with -d. When using debug

mode with a port reservation statement for the wrong job name, the bind() error can be ignored or the -i command-line option can be specified along with -d so that syslogd will not get a UDP socket.

Configuring TCPIP.DATA

Use of TCPIP.DATA and /etc/resolv.conf

The TCPIP.DATA configuration data set is the anchor configuration data set for the TCP/IP stack and all TCP/IP servers and clients running in z/OS. In z/OS IP, you can define the TCPIP.DATA parameters in an HFS file or in an MVS data set. The TCPIP.DATA configuration data set is read during initialization of all TCP/IP server and client functions. All functions must access this data set in order to find basic configuration information, such as the name of the TCP/IP address space, the TCP/IP host name, and the data set prefix to use when searching for other configuration data sets.

The TCPIP.DATA data set is also known as one of the resolver configuration data sets. In fact, this name is now more commonly used to refer to this important file in the UNIX System Services environment because the socket library contains a component called the resolver. In a UNIX system, you use the /etc/resolv.conf file for the same purpose as you use TCPIP.DATA in your MVS system.

TCPIP.DATA specifies the name of the TCP/IP address space. Because the data set search order can vary, your installation will determine which data set you can use. See Chapter 1, “Configuration overview” on page 3 for search order, data set, and file retrieval information and “Resolver configuration files” on page 27.

If you use TCPIP.DATA, it can be shared between multiple systems with a system name. But, if TCPIP.DATA is allocated via SYSTCPD DD and an application forks, any allocations from the parent of SYSTCPD are lost to the child process.

In z/OS UNIX System Services, each application can have its own environment variable, RESOLVER_CONFIG=’xxx’. There are no concerns for forked child processes; however, this means that you cannot share the same data set or file among multiple systems.

Creating TCPIP.DATA

Create a TCPIP.DATA file by copying the sample provided in SEZAINST(TCPDATA) and modifying it to suit your local conditions.

Allocate this data set with either sequential (PS) or partitioned (PO) organization, a fixed (F) or fixed block format (FB), a logical record length (LRECL) between 80 and 256, and any valid block size for a fixed block. This file can also be the HFS file /etc/resolv.conf, or an HFS file that is pointed to by either the environment variable RESOLVER_CONFIG or the SYSTCPD DD in a JCL procedure. If you have an HFS file, the maximum line length can be 256. The environment variable RESOLVER_CONFIG can also point to an MVS data set or PDS.

You can use any name for the TCPIP.DATA data set if you access it using the //SYSTCPD DD statement, or use ENVAR to set RESOLVER_CONFIG, in the JCL for all the servers, logon procedures, and batch jobs that execute TCP/IP functions. If you are not using the //SYSTCPD DD statement, the environment variable, or /etc/resolv.conf, then the data set name must conform to the conventions described in “Configuration files for the TCP/IP stack” on page 25. Another alternative is to use

the well-known data set name SYS1.TCPPARMS(TCPDATA). You will issue the HOMETEST command with TRACE RESOLVER activated to verify the actual data set name the system finds for TCPIP.DATA later in this chapter. However, because HOMETEST is an MVS sockets application, it does not use RESOLVER_CONFIG or /etc/resolv.conf in its search order. For this reason, it is recommended that /etc/resolv.conf and TCPIP.DATA contain exactly the same information or consider using the resolver GLOBALTCPIPDATA setup statement.

TCPIP.DATA statements

Each configuration statement can be preceded by an optional *system_name*. This permits configuration information for multiple systems to be specified in a single *hlq*.TCPIP.DATA data set. The *system_name* is matched against the name of the system on which you are running. The name of the system is taken from the name in the IEFSSNxx member of parmlib, which is the third parameter of the VMCF line.

The statements are processed in the order they appear in the data set. The following rules apply to this processing:

- If the *system_name* does not match the name of the system, the configuration statement is ignored.
- If *system_name* is blank, the configuration statement is in effect on every system.
- If the *system_name* matches the host's name, the configuration statement that follows it is in effect.
- The ***last*** statement that matches is effective.

For example, if you have the following three TCPIPJOBNAME statements, MVS6 would look for a TCP/IP cataloged procedure named TCPBTA2, MVSA would look for TCPV3, and all other systems would look for TCPMCWN.

```
      TCPIPJOBNAME TCPMCWN
MVS6: TCPIPJOBNAME TCPBTA2
MVSA: TCPIPJOBNAME TCPV3
```

But if you reversed the order, all systems would try to find the procedure named TCPMCWN.

```
MVS6: TCPIPJOBNAME TCPBTA2
MVSA: TCPIPJOBNAME TCPV3
      TCPIPJOBNAME TCPMCWN
```

A sample TCPIP.DATA data set (TCPDATA) can be found in SEZAINST. For detailed information on each of the statements, refer to the *z/OS Communications Server: IP Configuration Reference*.

Configuring PROFILE.TCPIP

During TCP/IP address space initialization, a configuration profile data set (PROFILE.TCPIP) is read that contains system operation and configuration parameters. A sample data set, SEZAINST(SAMPPROF), can be copied and modified for use as your default configuration profile.

If you are not familiar with the search order for this data set, see "PROFILE.TCPIP search order" on page 25 for information about understanding data set search orders. Refer to *z/OS Communications Server: IP Configuration Reference* for the complete statement syntax and descriptions of the configuration statements.

For ease of management when configuring a complex environment, you can use one of the following PROFILE.TCPIP data set features:

- Group related statements into separate files and use the INCLUDE statement in PROFILE.TCPIP to include them in your configuration.
- Use MVS system symbols (such as &SYSCLONE, &SYSNAME, and &SYSPLEX). Because TCP/IP translates these symbols as it reads this file, this feature reduces the number of PROFILE.TCPIP data sets that must be maintained in a multi-TCP/IP environment.

Note: For detailed information about symbols and how to define them, refer to *z/OS MVS Initialization and Tuning Reference*.

The PROFILE data set contains the following major groups of configuration parameters:

- TCP/IP operating characteristics
- TCP/IP physical characteristics
- TCP/IP reserved port number definitions (application configuration)
- TCP/IP network routing definitions
- TCP/IP diagnostic data statements

This chapter discusses the first three areas of configuration. For routing configuration information, see Chapter 4, “Routing” on page 155. For information about configuring diagnostic statements, see *z/OS Communications Server: IP Diagnosis*.

Changing configuration information

If you want to change the TCP/IP configuration without stopping and starting the TCP/IP address space, you can dynamically change many of the TCP/IP configuration options established by the PROFILE.TCPIP data set. To do this, put the changed configuration statements in a separate data set and process it with the VARY TCPIP,,OBEYFILE command.

For more information about VARY TCPIP, refer to *z/OS Communications Server: IP System Administrator's Commands*. Also, see the Modifying section in each configuration statement in *z/OS Communications Server: IP Configuration Reference* for a description of how to dynamically change the information for that configuration statement.

Note: If you attempt to edit PROFILE.TCPIP while TCPIP is active, and PROFILE.TCPIP is defined in the TCPIP PROC as a sequential data set (for example, //PROFILE DD DISP=SHR,DSNAME=TCPIP.PROFILE.TCPIP), the *Dataset in use* message might be displayed. To avoid this, specify FREE=CLOSE, as follows:

```
//PROFILE  DD DISP=SHR,DSNAME=TCPIP.PROFILE.TCPIP,FREE=CLOSE
```

This allows you to edit the profile while TCP/IP is active. Typically, when TCP/IP starts, it keeps the PROFILE allocated and does not release the allocation until the end of the step (in this case, the end of the job). If you specify FREE=CLOSE, the release occurs once the data set is read. MVS releases the enqueue on the PROFILE, which allows you to edit it.

If the PROFILE is a member of a PDS, [for example, SYS1.TCPPARMS(PROFILE)], FREE=CLOSE is not needed.

Setting up TCP/IP operating characteristics in PROFILE.TCPIP

Figure 26 shows a portion of the sample configuration file for the TCP/IP address space, PROFILE.TCPIP. This sample can be copied from SEZAINST(SAMPPROF). Figure 26 includes the portion of the sample that shows how to set up TCP/IP operating characteristics. Descriptions for the statements follow Figure 26. For more information about any of these statements, refer to *z/OS Communications Server: IP Configuration Reference*. For information specific to IPv6 support, refer to *z/OS Communications Server: IPv6 Network and Application Design Guide*.

Figure 26. Example of TCP/IP operating characteristics in PROFILE.TCPIP

```
; ====================================================================
; General TCP/IP address space configuration
; ====================================================================
;
; ARPAGE: Specifies the number of minutes between creation or
;   revalidation of an LCS ARP table entry and the deletion of the
;   entry.
;
ARPAGE 20
;
;
; GLOBALCONFIG: Provides settings for the entire TCP/IP stack
;
GLOBALCONFIG NOTCPIPSTATISTICS
;
; IPCONFIG: Provides settings for the IPv4 IP layer of TCP/IP.
;
; Example IPCONFIG for single stack/single system:
;
IPCONFIG DATAGRAMFWD VARSUBNETTING SYSPLEXROUTING
;
; Example IPCONFIG for automatic activation of inter-stack dynamic XCF
;   and Same Host (IUTSAMEH) links
;
; IPCONFIG DYNAMICXCF 201.1.10.10 255.255.255.0 2
;
; IPCONFIG6: Provides settings for the IPv6 IP layer of TCP/IP.
;
; IPCONFIG6 DATAGRAMFWD SOURCEVIPA
;
; SOMAXCONN: Specifies maximum length for the connection request queue
;   created by the socket call listen().
;
SOMAXCONN 10
;
;
; TCPCONFIG: Provides settings for the TCP layer of TCP/IP.
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to authorized applications.  Applications can be
;            authorized to low ports in three ways:
;             - via PORT or PORTRANGE with the appropriate jobname
;                or wildcard jobname
;             - APF authorized
;             - superuser
;
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
TCPCONFIG RESTRICTLOWPORTS
;
;
; UDPCONFIG: Provides settings for the UDP layer of TCP/IP
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to authorized applications.  Applications can be
```

```
;            authorized to low ports in three ways:
;             - via PORT or PORTRANGE with the appropriate jobname
;                or wildcard jobname
;             - APF authorized
;             - superuser
;
UDPCONFIG RESTRICTLOWPORTS
;
```

The following section explains the grouping of statements shown in Figure 26 on page 112.

ARPAGE

Use ARPAGE to set the number of minutes between a revalidation and deletion of ARP table entries for LCS devices. An installation that wants to describe this value in seconds versus minutes should use the IPCONFIG ARPTO statement.

Note: The ATM ARP requests are controlled via the ATMLIS statement, and the MPCIPA and MPCOSA ARP requests are not controlled by the TCP/IP address space.

GLOBALCONFIG

Use GLOBALCONFIG to print out several counters in text format. These counters include number of TCP retransmissions and total number of TCP segments sent from the TCPIP system. Most installations will use the SMF facility of MVS to collect these counters in a more standard way. Use the ECSALIMIT parameter on the GLOBALCONFIG statement to limit TCP/IP's use of common storage. The POOLLIMIT parameter can be used to limit TCP/IP's use of private storage pools.

IPCONFIG

Use IPCONFIG to configure various settings of the IP layer of TCP/IP. Use ARPTO to specify the ARP time out value in seconds for LCS devices. See page 113 for more information.

Use CLAWUSEDOUBLENOP on vendor devices that document the need for double NOPs on each CCW.

Use DATAGRAMFWD if this TCP/IP is to be a router and needs to forward datagrams to other routers. Use IGNOREREDIRECT when a dynamic routing program is used and ICMP redirect packets are to be ignored by the TCP/IP address space. MULTIPATH is used to inform TCP/IP how to distribute traffic across equal cost routes. VARSUBNETTING allows the TCP/IP routing table to have address routes within the same subnet that have differing subnet masks.

Use FIREWALL to restrict this host to be a network firewall. To make IPSEC tunnels associated with Dynamic VIPA addresses eligible for distribution, if the VIPA addresses are being distributed and are eligible to be moved during VIPA takeover or giveback, add the DVIPSEC keyword to FIREWALL.

SOURCEVIPA enables interface fault tolerance for z/OS clients that establish outbound connections. When SOURCEVIPA is set, outbound datagrams use the corresponding virtual IP address (VIPA) in the HOME list instead of the physical interfaces IP address. SOURCEVIPA has no effect on RIP servers such as OROUTED, NCPROUTE, or OMPROUTE.

TCPSTACKSOURCEVIPA allows z/OS clients to specify a sysplex wide source IP address for TCP connections. When TCPSTACKSOURCEVIPA is

set, outbound TCP datagrams use the IP address specified in the TCPSTACKSOURCEVIPA statement instead of static VIPA addresses or physical interface addresses.

Use SYSPLEXRouting to communicate interface changes within a sysplex domain to the workload manager (WLM). DYNAMICXCF allows the cross communication facility within a sysplex to dynamically generate connections within a sysplex domain. If DYNAMICXCF is used with a routing program like OMPROUTE or OROUTED, then the BSDROUTINGPARMS and the OMPROUTE configuration files need to be updated with subnet mask and cost information. For more information on additional configuration parameters required, see the usage notes related to the DYNAMICXCF parameter under the IPCONFIG statement in *z/OS Communications Server: IP Configuration Reference*.

Use REASSEMBLYTIMEOUT to specify the TCP/IP reassemble timeout value in seconds, and the TTL specifies the TCP/IP time to live or hop count value.

Use PATHMTUDISCOVERY to indicate to TCP/IP that it is to dynamically discover the path MTU, which is the minimum of MTUs of each hop in the path.

Use STOPONCLAWERROR to indicate to the TCP/IP stack to stop channel programs (HALTIO and HALTSIO) when a device error is detected.

IPCONFIG6

Use IPCONFIG6 to update the IP layer of TCP/IP with information that pertains to IPv6. Use DATAGRAMFWD to enable the transfer of data between networks.

SOMAXCONN

Use SOMAXCON to specify the maximum number of sockets queued on a listener.

TCPCONFIG

Use TCPCONFIG to configure various settings of the TCP protocol layer. If a keep-alive value other than 120 minutes is needed by an installation, use the INTERVAL statement to change the default keep-alive value. FINWAIT2TIME can be used to specify a different timeout value for a TCP Connection which is in a FINWAIT2 state. SENDGARBAGE will cause the keep-alive packet to contain one byte of random data and an incorrect sequence number, assuring that the data is not accepted by the remote TCP. The TCPTIMESTAMP option can be used to choose whether or not to participate in timestamp negotiation.

The behavior of acknowledgments and delaying their transmission can be altered by using the DELAYACKS statement.

If RESTRICTLOWPORTS is specified, only applications that meet at least one of the following criteria are allowed to bind to low ports (1–1023):

- The port is reserved for the application via the PORT or PORTRANGE statement.
- The application runs with APF authorization.
- The application runs with effective POSIX UID zero.

If an installation wants to control TCP buffering (to limit storage usage or to manage large bandwidth devices), use the TCPSENDBFRSIZE, TCPRCVBUFRSIZE, and TCPMAXRCVBUFRSIZE parameters.

UDPCONFIG

Use UDPCONFIG to configure various settings of the UDP protocol layer. NOUDPCHKSUM can be used to eliminate check summing overhead for IPv4 UDP packets. This option is ignored for UDP datagrams flowing over an IPv6 network, as UDP Checksum is a required function on an IPv6 network.

If RESTRICTLOWPORTS is specified, only applications that meet at least one of the following criteria are allowed to bind to low ports (1–1023):

- The port is reserved for the application via the PORT or PORTRANGE statement.
- The application runs with APF authorization.
- The application runs with effective POSIX UID zero.

If an installation wants to control UDP buffering (to limit storage usage or to manage large bandwidth devices), use the UDPSENDBFRSIZE and UDPRCVBUFRSIZE parameters. UDPQUEUELIMIT can be used to set a queue limit for UDP. This is useful for installations that want to limit the size of the queue of UDP datagrams that an application can have waiting before the TCP/IP address space starts discarding them.

Setting up physical characteristics in PROFILE.TCPIP

Figure 27 shows the sample configuration file for the TCP/IP address space, PROFILE.TCPIP. This sample can be copied from SEZAINST(SAMPPROF). Following Figure 27, several of the statements that are used to set up physical characteristics in PROFILE.TCPIP are described. For more information about any of these statements, or information on statements not described, refer to *z/OS Communications Server: IP Configuration Reference*. For information specific to IPv6 support, refer to *z/OS Communications Server: IPv6 Network and Application Design Guide*.

Figure 27. Example of physical characteristics in PROFILE.TCPIP

```
;
; PROFILE.TCPIP
; =============
;
; This is a sample configuration file for the TCPIP address space
;
; SMP/E name: EZAEB025, alias SAMPPROF in target library SEZAINST
;
; COPYRIGHT = NONE
;
; Notes:
;
; - The device configuration, home and routing statements MUST be
;   changed to match your hardware and software configuration.
;   Likewise, the BEGINVTAM section MUST be changed to match your
;   VTAM configuration.
;
; - Lines beginning with semi-colons are comments.  To use a line
;   for your configuration, remove the semi-colon.
;
; - For more information about this file, see the IP Configuration Guide
;
; ====================================================================
; General TCP/IP address space configuration
; ====================================================================
;
; ARPAGE: Specifies the number of minutes between creation or
;   revalidation of an LCS ARP table entry and the deletion of the
;   entry.
```

```
;
ARPAGE 20
;
;
; GLOBALCONFIG: Provides settings for the entire TCP/IP stack
;
GLOBALCONFIG NOTCPIPSTATISTICS
;
; IPCONFIG: Provides settings for the IPv4 IP layer of TCP/IP.
;
; Example IPCONFIG for single stack/single system:
;
IPCONFIG DATAGRAMFWD VARSUBNETTING SYSPLEXROUTING
;
; Example IPCONFIG for automatic activation of inter-stack dynamic XCF
;   and Same Host (IUTSAMEH) links
;
; IPCONFIG DYNAMICXCF 201.1.10.10 255.255.255.0 2
;
; IPCONFIG6: Provides settings for the IPv6 IP layer of TCP/IP.
;
; IPCONFIG6 DATAGRAMFWD SOURCEVIPA
;
; SOMAXCONN: Specifies maximum length for the connection request queue
;   created by the socket call listen().
;
SOMAXCONN 10
;
;
; TCPCONFIG: Provides settings for the TCP layer of TCP/IP.
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to authorized applications.  Applications can be
;            authorized to low ports in three ways:
;             - via PORT or PORTRANGE with the appropriate jobname
;                or wildcard jobname
;             - APF authorized
;             - superuser
;
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
TCPCONFIG RESTRICTLOWPORTS
;
;
; UDPCONFIG: Provides settings for the UDP layer of TCP/IP
;            RESTRICTLOWPORTS limits access to ports below 1024
;            to authorized applications.  Applications can be
;            authorized to low ports in three ways:
;             - via PORT or PORTRANGE with the appropriate jobname
;                or wildcard jobname
;             - APF authorized
;             - superuser
;
UDPCONFIG RESTRICTLOWPORTS
;
;
; ====================================================================
; Hardware definitions
; ====================================================================
;
; DEVICE: Defines name (and sometimes device number) for various types
;   of network devices for IPv4 only
; LINK: Defines a network interface to be associated with a particular
;   device.  For IPv4 only.
; INTERFACE: Defines an IPv6 interface.
;
;
; DEVICE and LINK for CTC devices
;
;DEVICE  CTC1    CTC  D00  AUTORESTART
;LINK    CTCD00  CTC  0  CTC1
;
; DEVICE and LINK for HYPERchannel A220 devices:
;
;DEVICE  HCH1    HCH  E00  AUTORESTART
;LINK    HCHE00  HCH  1  HCH1
;
; DEVICE and LINK for LAN Channel Station and OSA devices:
```

```
;   DEVICE: Defines name and hexadecimal device number for an IBM 8232
;     LAN channel station (LCS) device, and IBM 3172 Interconnect
;     Controller, an IBM 2216 Multiaccess Connector Model 400,
;     an IBM FDDI, Ethernet, or Token Ring OSA, or an IBM ATM OSA-2
;     in LAN emulation mode
;   LINK: Defines a network interface link associated with an LCS
;     device; may be for Ethernet Network, Token-Ring Network or
;     PC Network, or FDDI.
;
;   Example: LCS1 is a 3172 model 1 with a Token Ring and Ethernet
;    adapter
;
;DEVICE  LCS1  LCS  BA0  AUTORESTART
;LINK    TR1   IBMTR  0  LCS1
;LINK    ETH1  ETHERNET  1  LCS1
;
;   Example: LCS2 is a 3172 model 2 with a FDDI adapter
;
;DEVICE  LCS2   LCS  BE0  AUTORESTART
;LINK    FDDI1  FDDI  0  LCS2
;
; DEVICE and LINK for MPCIPA QDIO Devices:
;
;   Example: MPCIPA1 is either an IBM OSA-Express Gigabit Ethernet
;    or QDIO Fast Ethernet adapter
;
;DEVICE  MPCIPA1     MPCIPA  NONROUTER AUTORESTART
;LINK    MPCIPALINK1 IPAQENET  MPCIPA1
;
;   Example: MPCIPA2 is either an IBM OSA-Express Gigabit Ethernet
;    or QDIO Fast Ethernet adapter, configured as the PRIMARY router
;
;DEVICE  MPCIPA2     MPCIPA  PRIROUTER AUTORESTART
;LINK    MPCIPALINK2 IPAQENET  MPCIPA2
;
;   DEVICE and LINK for MPCPTP devices:
;
;DEVICE  MPCPTP1    MPCPTP   AUTORESTART
;LINK    MPCPTPLINK MPCPTP   MPCPTP1
;
;    DEVICE and LINK for CLAW devices:
;
;DEVICE  RS6K    CLAW  6B2  HOST  PSCA  NONE  26 26 AUTORESTART
;LINK    IPLINK1 IP  0  RS6K
;
;   DEVICE and LINK for SNA LU0 links:
;
;DEVICE  SNALU0  SNAIUCV   SNALINK  LU000000  SNALINK  AUTORESTART
;LINK    SNA1    SAMEHOST  1  SNALU0
;
;    DEVICE and LINK for SNA LU 6.2 links:
;
;DEVICE  SNALU621 SNALU62 SNAPROC AUTORESTART
;LINK    SNA2     SAMEHOST  1  SNALU621
;
;    DEVICE and LINK for X.25 NPSI connections:
;
;DEVICE  X25DEV   X25NPSI  TCPIPX25  AUTORESTART
;LINK    X25LINK  SAMEHOST  1  X25DEV
;
;    DEVICE and LINK for 3745/46 Channel DLC Devices:
;
;DEVICE  CDLC1    CDLC  C00 AUTORESTART
;LINK    CDLCLINK CDLC  1  CDLC1
;
;    DEVICE and LINK for MPC OSA Fast Ethernet Devices:
;
;DEVICE  MENET1   MPCOSA   AUTORESTART
;LINK    ENETLINK OSAENET 0 MENET1
;
;    DEVICE and LINK for MPC OSA FDDI Devices:
;
;DEVICE  MFDDI1   MPCOSA   AUTORESTART
;LINK    FDDILINK OSAFDDI 0 MFDDI1
;
; ---------------------------------------------------------------------
```

```
; Virtual device definitions
; -------------------------------------------------------------------
;
; DEVICE and LINK for Virtual Devices (VIPA):
;
;   DEVICE  VDEV1   VIRTUAL  0
;   LINK    VLINK1  VIRTUAL  0  VDEV1
;
; Dynamic Virtual Devices can be defined on this system.  This system
; can serve as backup for Dynamic Virtual Devices on other systems.
; A predefined range will allow Dynamic Virtual Devices to be defined
; by IOCTL or Bind requests.
;
; VIPADYNAMIC
;   Define two dynamic VIPAs on this stack:
;   VIPADEFINE 255.255.255.192 201.2.10.11 201.2.10.12
;
;   Define this stack as backup for these dynamic VIPAs on
;   other TCP/IP stacks:
;   VIPABACKUP 100             201.2.10.13 201.2.10.14
;   VIPABACKUP  80             201.2.10.21 201.2.10.22
;   VIPABACKUP  60             201.2.10.31 201.2.10.33
;   VIPABACKUP  40             201.2.10.32 201.2.10.34
;
;   VIPARANGE DEFINE 255.255.255.192 201.2.10.192
; ENDVIPADYNAMIC
;
; -------------------------------------------------------------------
; ATM hardware definitions
; -------------------------------------------------------------------
;
; ATMLIS: Describes characteristics of an ATM logical IP subnet (LIS).
;
; DEVICE and LINK for ATM devices: (See below)
;
; ATMPVC: Describes a permanent virtual circuit (PVC) to be used by an
;   ATM link.
;
; ATMARPSV: Designates the ATMARP server that will resolve ATMARP
;   requests for a logical IP subnet (LIS).
;
; ATMLIS   LIS1   9.67.100.0 255.255.255.0
; DEVICE   OSA1   ATM  PORTNAME  PORT1
; LINK     LINK1  ATM  OSA1  LIS LIS1
; ATMPVC   PVC1   LINK1
; ATMARPSV ARPSV1 LIS1 PVC PVC1
;
;
; -------------------------------------------------------------------
; IPv6 INTERFACE statements
; -------------------------------------------------------------------
;    Virtual interface definitions
;      IPADDR keyword is required for Virtual interfaces
;      Multiple IP addresses can be defined to one interface
;      The prefixes of the IPv6 VIPA addresses should be
;        different than the prefixes used for addresses
;        configured or autoconfigured for real interfaces.
; -------------------------------------------------------------------
;  INTERFACE VIPAV6 DEFINE
;    VIRTUAL6
;    IPADDR FEC0:0:0:A:9:67:115:66    ; (Site-Local Address)
;           50C9:C2D4:0:A:9:67:115:66 ; (Global Address)
; -------------------------------------------------------------------
; To use autoconfiguration, the IPADDR cannot be specified.
; To manally define address(es), use the IPADDR keyword.
; To assign a VIPA address for an interface, use SOURCEVIPAINT
; To have IPv4 and IPv6 share a physical device, define IPv4
;   using DEVICE/LINK/HOME and IPv6 using INTERFACE
; -------------------------------------------------------------------
;  INTERFACE OSAQDIO26                ; OSA QDIO (Fast Ethernet)
;    DEFINE IPAQENET6
;    PORTNAME OSAQDIO2
;    SOURCEVIPAINT VIPAV6
;    IPADDR FEC0:0:0:1:9:67:115:66    ; (Site-Local Address)
;           50C9:C2D4:0:1:9:67:115:66 ; (Global Address)
; -------------------------------------------------------------------
```

```
; To define other Ipv6 Loopback addresses:
; ----------------------------------------------------------------------
;   INTERFACE LOOPBACK6 ADDADDR ::0014:0
; ----------------------------------------------------------------------
; Other device statements
; ----------------------------------------------------------------------
;
;
; TRANSLATE: Indicates a relationship between an internet address and
;   the network address on a specified link.  Only applicable for IPv4
;   devices.
;
; TRANSLATE
;   9.67.43.110   FDDI    FF0000006702   FDDI1
;   9.37.84.49    HCH     FF0000005555   HCHE00
;
;
; ======================================================================
; HOME addresses
; ======================================================================
;
; HOME: Provides the list of home IP addresses and associated link names
;       for IPv4
;
;   - The LOOPBACK statement of 14.0.0.0 should only be used if the
;     installation has applications that require this old loopback
;     address.  The current stack uses 127.0.0.1 as the loopback
;     address.
;
; HOME
;  14.0.0.0       LOOPBACK
;  130.50.75.1    TR1
;  193.5.2.1      ETH1
;  9.67.43.110    FDDI1
;  193.7.2.1      SNA1
;  9.67.113.80    CTCD00
;  9.37.84.49     HCHE00
;  9.67.113.81    MPCIPALINK1
;  9.67.113.82    MPCPTPLINK
;  9.67.113.83    MPCIPALINK2
;  9.67.114.02    IPLINK1
;  9.67.43.03     SNA2
;  9.67.115.85    X25LINK
;  9.67.116.86    VLINK1
;  9.67.117.87    CDLCLINK
;  9.67.100.80    LINK1
;  9.37.112.13    ENETLINK
;  9.37.112.14    FDDILINK
;
;
; PRIMARYINTERFACE: Specifies which link is designated as the default
;   local host for use by the GETHOSTID() function.  Only applicable
;   for IPv4 devices.
;
;   - If PRIMARYINTERFACE is not specified, then the first link in
;     the HOME statement is the primary interface, as usual.
;
; PRIMARYINTERFACE TR1
;
; ======================================================================
; Routing configuration
; ======================================================================
; ----------------------------------------------------------------------
; Static routing
; ----------------------------------------------------------------------
;
; BEGINRoutes: Defines static routes to the IP route table for IPv4
;                  and IPv6
;
BEGINRoutes
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
;      Destination Subnet Mask   First Hop    Link Name Packet Size
;
;ROUTE 130.50.75.0 255.255.255.0 =            TR1       MTU 2000
```

```
;ROUTE 193.5.2.0/24              =            ETH1      MTU 1500
;ROUTE 9.67.43.0   255.255.255.0 =            FDDI1     MTU 4000
;ROUTE 193.7.2.2   HOST          =            SNA1      MTU 2000
;
;      Destination Subnet Mask   First Hop  Interface   Packet Size
;
;  ROUTE FE80::1:2:3:4/128         =        OSAQDIO26     MTU 2000
;  ROUTE FEC0::1/128               =        OSAQDIO26     MTU 2000
;
;
; Indirect Routes - Routes that are reachable through routers on my
;                   network.
;
;      Destination Subnet Mask   First Hop    Link Name Packet Size
;
;ROUTE 193.12.2.0  255.255.255.0 130.50.75.10 TR1       MTU 2000
;ROUTE 10.5.6.4    HOST          193.5.2.10   ETH1      MTU 1500
;
;      Destination Subnet Mask   First Hop  Interface   Packet Size
;
;  ROUTE FEC8::/64           FE80::1:2:3:4    OSAQDIO26   MTU 2000
;
; Default Route - All packets to an unknown destination are routed
;                 through this route.
;
;      Destination               First Hop    Link Name Packet Size
;
;ROUTE DEFAULT                   9.67.43.99   FDDI1     MTU DEFAULTSIZE
;
;      Destination Subnet Mask   First Hop  Interface   Packet Size
;
; ROUTE DEFAULT6             FE80::1:2:3:4  OSAQDIO26   MTU DEFAULTSIZE
ENDRoutes
;
; ----------------------------------------------------------------------
; Dynamic routing
;    Only support for IPv4 at this time.
; ----------------------------------------------------------------------
;
; BSDROUTINGPARMS: Defines the characteristics of each link defined at
;   the host over which OROUTED will send routing information to
;   adjacent routers running the RIP protocl and which NCPROUTE will
;   send transport PDUs to client NCPs.
;
;   - OMPROUTE is the recommended routing daemon.  It does not use
;     BSDROUTINGPARMS.
;
;   - OROUTED users must define BSDROUTINGPARMS.
;
;   - Use of the BEGINROUTES statement (static routes) with the
;     OMPROUTE or OROUTED routing daemons is not recommended.
;
; BSDROUTINGPARMS TRUE
;  Link name      MTU    Cost metric  Subnet Mask    Dest address
;   TR1          2000         0       255.255.255.0     0
;   ETH1         1500         0       255.255.255.0     0
;   FDDI1        4000         0       255.255.255.0     0
;   VLINK1    DEFAULTSIZE     0       255.255.255.0     0
;   CTCD00       65527        0       255.255.255.0  9.67.113.90
; ENDBSDROUTINGPARMS
;
;
; ======================================================================
; Application configuration
; ======================================================================
;
;
; AUTOLOG: Supplies TCPIP with the procedure names to start and the
; time value to wait at TCP start up for any of those procedures
; to terminate if they are active.
;
; AUTOLOG 5
;  FTPD JOBNAME FTPD1       ; FTP Server
;  LPSERVE                  ; LPD Server
;  NAMED                    ; Domain Name Server
;  NCPROUT                  ; NCPROUTE Server
```

```
;  OROUTED                 ; OROUTED Server
;  OSNMPD                  ; SNMP Agent Server
;  PORTMAP                 ; Portmap Server (SUN 3.9)
;  PORTMAP JOBNAME PORTMAP1 ; USS Portmap Server (SUN 4.0)
;  RXSERVE                 ; Remote Execution Server
;  SMTP                    ; SMTP Server
;  SNMPQE                  ; SNMP Client
;  TCPIPX25                ; X25 Server
; ENDAUTOLOG
;
;
; PORT: Reserves a port for specified job names
;
;   - A port that is not reserved in this list can be used by any user.
;     If you have TCP/IP hosts in your network that reserve ports
;     in the range 1-1023 for privileged applications, you should
;     reserve them here to prevent users from using them.
;     The RESTRICTLOWPORTS option on TCPCONFIG and UDPCONFIG will also
;     prevent unauthorized applications from accessing unreserved
;     ports in the 1-1023 range.
;
;   - A PORT statement with the optional keyword SAF followed by a
;     1-8 character name can be used to reserve a PORT and control
;     access to the PORT with a security product such as RACF.
;     For port access control, the full resource name for the security
;     product authorization check is constructed as follows:
;     EZB.PORTACCESS.sysname.tcpname.safname
;     where:
;       EZB.PORTACCESS is a constant
;       sysname is the MVS system name (substitute your sysname)
;       tcpname is the TCPIP jobname (substitute your jobname)
;       safname is the 1-8 character name following the SAF keyword
;
;     When PORT access control is used, the TCP/IP application
;     requiring access to the reserved PORT must be running under a
;     USERID that is authorized to the resource. The resources
;     are defined in the SERVAUTH class.
;
;     For an example of how the SAF keyword can be used to enhance
;     security, see the definition below for the FTP data PORT 20
;     with the SAF keyword. This definition reserves TCP PORT 20 for
;     any jobname (the *) but requires that the FTP user be permitted
;     by the security product to the resource:
;     EZB.PORTACCESS.sysname.tcpname.FTPDATA in the SERVAUTH class.
;
;   - The BIND keyword is used to force a generic server (one that
;     binds to INADDR_ANY) to bind to the specific IP address that
;     is specified following the BIND keyword. This capability could
;     be used, for example, to allow z/OS UNIX telnet and telnet
;     3270 servers to both bind to TCP port 23.
;     The IP address that follows bind must be in IPv4 dotted
;     decimal format and may be any valid address for the host
;     including VIPA and dynamic VIPA addresses.
;
;   The special jobname of OMVS indicates that the PORT is reserved
;   for any application with the exception of those that use the Pascal
;   API.
;
;   The special jobname of * indicates that the PORT is reserved
;   for any application, including Pascal API socket applications.
;
;   The special jobname of RESERVED indicates that the PORT is
;   blocked. It will not be available to any application.
;
;   The special jobname of INTCLIEN indicates that the PORT is
;   reserved for internal stack use.
;
;
PORT
     7 UDP MISCSERV           ; Miscellaneous Server - echo
     7 TCP MISCSERV           ; Miscellaneous Server - echo
     9 UDP MISCSERV           ; Miscellaneous Server - discard
     9 TCP MISCSERV           ; Miscellaneous Server - discard
    19 UDP MISCSERV           ; Miscellaneous Server - chargen
    19 TCP MISCSERV           ; Miscellaneous Server - chargen
    20 TCP *  NOAUTOLOG       ; FTP Server
```

```
;   20 TCP *  NOAUTOLOG SAF FTPDATA ; FTP Server
    21 TCP FTPD1               ; FTP Server
;   21 TCP FTPD2 BIND FEC9:C2D4:1:0000:0009:0067:0115:0066 ; FTP IPv6
    23 TCP INTCLIEN            ; Telnet 3270 Server
;   23 TCP INETD1 BIND 9.67.113.3 ; z/OS UNIX Telnet server
    25 TCP SMTP                ; SMTP Server
    53 TCP NAMED               ; Domain Name Server
    53 UDP NAMED               ; Domain Name Server
   111 TCP PORTMAP             ; Portmap Server (SUN 3.9)
   111 UDP PORTMAP             ; Portmap Server (SUN 3.9)
;  111 TCP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
;  111 UDP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
   123 UDP SNTPD               ; Simple Network Time Protocol Server
   135 UDP LLBD                ; NCS Location Broker
   161 UDP OSNMPD              ; SNMP Agent
   162 UDP SNMPQE              ; SNMP Query Engine
   389 TCP LDAPSRV             ; LDAP Server
   443 TCP HTTPS               ; http protocol over TLS/SSL
   443 UDP HTTPS               ; http protocol over TLS/SSL
   512 TCP RXSERVE             ; Remote Execution Server
   514 TCP RXSERVE             ; Remote Execution Server
;  512 TCP * SAF OREXECD       ; z/OS UNIX Remote Execution Server
;  514 TCP * SAF ORSHELLD      ; z/OS UNIX Remote Shell Server
   515 TCP LPSERVE             ; LPD Server
   520 UDP OROUTED             ; OROUTED Server
   580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERB             ; Kerberos
   750 UDP MVSKERB             ; Kerberos
   751 TCP ADM@SRV             ; Kerberos Admin Server
   751 UDP ADM@SRV             ; Kerberos Admin Server
  1933 TCP ILMTSRVR            ; IBM LM MT Agent
  1934 TCP ILMTSRVR            ; IBM LM Appl Agent
  3000 TCP CICSTCP             ; CICS Socket
  3389 TCP MSYSLDAP            ; LDAP Server for Msys
;
;
; PORTRANGE: Reserves a range of ports for specified jobnames.
;
;   In a common INET (CINET) environment, the port range indicated by
;   the INADDRANYPORT and INADDRANYCOUNT in your BPXPRMxx parmlib member
;   should be reserved for OMVS.
;
;   The special jobname of OMVS indicates that the PORTRANGE is reserved
;   for ANY z/OS UNIX socket application.
;
;   The special jobname of * indicates that the PORTRANGE is reserved
;   for any socket application, including Pascal API socket
;   applications.
;
;   The special jobname of RESERVED indicates that the PORTRANGE is
;   blocked. It will not be available to any application.
;
;   The SAF keyword is used to restrict access to the PORTRANGE to
;   authorized users. See the use of SAF on the PORT statement above.
;
;
;   PORTRANGE 4000 1000 TCP OMVS
;   PORTRANGE 4000 1000 UDP OMVS
;   PORTRANGE 2000 3000 TCP RESERVED
;   PORTRANGE 5000 6000 TCP * SAF RANGE1
;
; SACONFIG: Configures the TCP/IP SNMP subagent
;
SACONFIG ENABLED COMMUNITY public AGENT 161
;
;
; ----------------------------------------------------------------------
; Configure Telnet
; ----------------------------------------------------------------------
;
; TELNETPARMS: Configure the Telnet Server
;
; - TN3270(E) server port 23 options
;
TelnetParms
  Port 23                      ; Port number 23 (std.)
```

```
  TELNETDEVICE 3278-3-E NSX32703 ; 32 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3279-3-E NSX32703 ; 32 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3278-4-E NSX32704 ; 48 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3279-4-E NSX32704 ; 48 line screen -
                                 ; default of NSX32702 is 24
  TELNETDEVICE 3278-5-E NSX32705 ; 132 column screen-
                                 ; default of NSX32702 is 80
  TELNETDEVICE 3279-5-E NSX32705 ; 132 column screen -
                                 ; default of NSX32702 is 80
  LUSESSIONPEND     ; On termination of a Telnet server connection,
                    ; the user will revert to the DEFAULTAPPL
                    ; instead of having the connection dropped

  MSG07             ; Sends a USS error message to the client if an
                    ; error occurs during session establishment
                    ; instead of dropping the connection
  CodePage ISO8859-1 IBM-1047   ; Linemode ASCII, EBCDIC code pages
  Inactive 0                    ; Let connections stay around
  PrtInactive 0                 ; Let connections stay around
  TimeMark 600
  ScanInterval 120
; SMFinit std
; SMFterm std
  WLMClusterName
    TN3270E
  EndWLMClusterName
  ; Define logon mode tables to be the defaults shipped with the
  ; latest level of VTAM

EndTelnetParms
;
; TelnetParms
;   Secureport 992 Keyring HFS /tmp/telnet.kdb
; EndTelnetParms
;
; BEGINVTAM: Defines the VTAM parameters required for the Telnet server.
;
BeginVTAM
  Port 23 ; 992
  ; Define the LUs to be used for general users.
  DEFAULTLUS
    TCPABC01..TCPABC99..FFFFFFNN
  ENDDEFAULTLUS
  DEFAULTAPPL TSO   ; Set the default application for all TN3270(E)
                    ; Telnet sessions to TSO

  LINEMODEAPPL TSO  ; Send all line-mode terminals directly to TSO.

; ALLOWAPPL SAMON QSESSION  ; SAMON appl does CLSDST Pass to next appl

  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
              ; applications.
              ; TSO is multiple applications all beginning with TSO,
              ; so use the * to get them all.  If a session is closed,
              ; disconnect the user rather than log off the user.

  ALLOWAPPL *       ; Allow all applications that have not been
                    ; previously specified to be accessed.

  RESTRICTAPPL IMS ; Only 3 users can use IMS.
    USER USER1      ; Allow user1 access.
      LU TCPIMS01   ; Assign USER1 LU TCPIMS01.
    USER USER2      ; Allow user2 access from the default LU pool.
    USER USER3      ; Allow user3 access from 3 Telnet sessions,
                    ; each with a different reserved LU.
      LU TCPIMS31 LU TCPIMS32 LU TCPIMS33

  ;   Map Telnet sessions from IP address 130.50.10.1 to display the
  ;   USSMSG10 screen from USS table USSAPC.

; USSTCP USSAPC 130.50.10.1

  ;   Map Telnet sessions from the SNA1 link to display the USSMSG10
```

```
  ;   screen from USS table USSCBA.

; USSTCP USSCBA SNA1

; LUGROUP  LUGRP1
;       TCPM0001..TCPM0999
;       TCPM1001
; ENDLUGROUP

; LUGROUP LUGRP2
;       TCPM2001  TCPM2003 TCPM2004
;       TCPM0AAA..TCPM0ZZZ
; ENDLUGROUP

  ; Define groups of host names
; HNGROUP HNGRP1
;      TEST1.TCP.RALEIGH.IBM.COM
;      TEST2.TCP.RALEIGH.IBM.COM
;      *.*.RALEIGH.IBM.COM
; ENDHNGROUP

; HNGROUP HNGRPALL
;      **.COM
; ENDHNGROUP

  ; Map LUs to groups for host names
; LUMAP LUGRP1 HNGRP1
; LUMAP LUGRP2 HNGRPALL
; LUMAP TCPM5000  SPECIAL.TCP.RALEIGH.IBM.COM
EndVTAM
;
;
; ----------------------------------------------------------------------
; Configure Network Access Control
; ----------------------------------------------------------------------
;     Network access contol can be used to restrict the destinations
;     that TCP/IP users are allowed to communicate with.
;
;     The NETACCESS group contains optional flags that control whether
;     checking is done on inbound paths (accept and all read variants)
;     and outbound paths (connect and all write variants).  If no flags
;     are specified, checking is only performed on outbound paths.
;     NOINBOUND and NOOUTBOUND disable checking in that direction.
;
;     The NETACCESS group also contains a list of IP addresses
;     that may be subnetworks or specific hosts.  The subnetwork mask
;     can be specified as a number of significant bits or in dotted
;     decimal notation.  The mask must be contiguous bits.
;
;     The special IP address DEFAULT with a mask of 0 includes all
;     IPv4 addresses not otherwise specified.
;
;     A 1-8 character name follows the IP address and subnet mask and
;     is used as the right-most qualifier in the security product
;     resource name.
;
;     For network access control, the full resource name for the
;     security product authorization check is constructed as follows:
;
;     EZB.NETACCESS.sysname.tcpname.resname
;     where:
;       EZB.NETACCESS is a constant
;       sysname is the MVS system name (substitute your sysname)
;       tcpname is the TCPIP jobname (substitute your jobname)
;       resname is the 1-8 character name following the subnet mask.
;
;     When network access control is used, the TCP/IP application
;     requiring access to the restricted subnet or host must be running
;     under a USERID that is authorized to the resource. The resources
;     are defined in the SERVAUTH class. See the EZARACF sample for
;     examples of the RACF definitions.
;
;NETACCESS    INBOUND    OUTBOUND           ; check both ways
;  192.168.0.0/16                   CORPNET ; Net address
;  192.168.113.19/32                HOST1   ; Specific host address
;  192.168.113.0     255.255.255.0  SUBNET1 ; Subnet address
```

```
;  192.168.112.0     255.255.248.0   SUBNET2 ; Subnet address
;  192.168.192.0/24                  CAMPUS  ; Subnet address
;  192.168.214.0/24                  CAMPUS  ; Subnet address
;  DEFAULT           0               DEFZONE ; Optional Default zone
;ENDNETACCESS
;
;
;
; ====================================================================
; Diagnostic data statements
; ====================================================================
;
; - For optimum performance, use of tracing should be limited to when
;   required for problem analysis.
;
; ITRACE: Controls TCP/IP run-time tracing
;
; ITRACE ON CONFIG 1
; ITRACE OFF SUBAGENT
;
;
; PKTTRACE: Controls the packet trace facility in TCP/IP.
;
; PKTTRACE ABBREV=200 LINKNAME=TR1 PROT=ICMP IP=*
;   SRCPORT=5000 DESTPORT=161
;
;
; SMFCONFIG: Provides SMF logging for Telnet, FTP, TCP API and TCP
;   stack activity.
;
;     - The SMF record types for TCP/IP records are 118 and 119.
;
; For Type 118 records specify:
;
; SMFCONFIG TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
;
; For Type 119 records specify:
;
; SMFCONFIG
;   TYPE119 TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
;           IFSTATISTICS PORTSTATISTICS TCPSTACK UDPTERM
;
; For all Type 118 and Type 119 records specify:
;
; SMFCONFIG TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
;   TYPE119 TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
;           IFSTATISTICS PORTSTATISTICS TCPSTACK UDPTERM
;
;
; SMFPARMS: Logs the use of TCP by applications using SMF log records.
;   However, use of the SMFCONFIG statement is recommended instead.
;
;
; ====================================================================
; Other statements
; ====================================================================
;
; DELETE: Removes an ATMARPSV, ATMLIS, ATMPVC, device, link, port or
;   portrange.  This statement is typically done via an obey file, not
;   in an initial profile.
;
; STOP: Stops a device.  If used, this statement is typically put in
;   an obey file, not in an initial profile.
;
; INCLUDE: Causes another data set that contains profile configuration
;   statements to be included at this point.
; --------------------------------------------------------------------
;
; START: Starts a device or interface that is currently stopped.
;
;START LCS1
;START LCS2
;START OSAQDIO26
```

The following section explains several of the statements shown in Figure 27 on page 115 that are used to set up physical characteristics in PROFILE.TCPIP. For more information about any of these statements, or information on statements not described here, see *z/OS Communications Server: IP Configuration Reference*. For information specific to IPv6 support, refer to *z/OS Communications Server: IPv6 Network and Application Design Guide*.

DEVICE and LINK

Use DEVICE and LINK statements to define each IPv4 network interface to the TCPIP address space. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details about the various network interfaces supported by TCP/IP.

ATM Use the ATM DEVICE and LINK statements to define connectivity to an ATM network. These statements allow for connectivity in either ATM native mode over an ATM virtual circuit (VC) or in ATM LAN Emulation mode.

For ATM native mode, the VC can be either a permanent virtual circuit (PVC) or a switched virtual circuit (SVC). To define a PVC, use the ATMPVC statement. To define SVCs, use the ATMLIS statement to define the ATM logical IP subnet (LIS). Also, for SVCs, use the ATMARPSV statement to define the ATMARP server that will resolve ATMARP requests within the LIS. For ATM LAN emulation mode, the ATM DEVICE and LINK definitions allow you to retrieve SNMP network management data for the device. In this mode, you need to define the device as an LCS.

CDLC The DEVICE CDLC describes the interface between the TCP/IP address space and the 3745/46 devices used.

CLAW Use CLAW DEVICE for RISC System/6000® and SP2®.

CTC Use the CTC DEVICE and LINK statements to define connectivity to another z/OS using channel-to-channel.

HYPERchannel A220 DEVICE and LINK

Use the HCH DEVICE and LINK statements to define connectivity via the HYPERchannel A220 adapter.

LAN Channel Station (LCS) DEVICE and LINK

Use the LCS DEVICE and LINK statements to define connectivity to a token-ring, FDDI, or Ethernet LAN. LCS devices can have more than one adapter. Therefore, you can have more than one LINK statement for an LCS DEVICE statement.

In configurations where multiple LCS and/or MPCIPA links onto the same LAN are defined, if the interface targeted by the ARP Request is inactive, one of the other active interfaces on the LAN will automatically take over responsibility for answering ARPs on behalf of the inactive interface. In this way, fault tolerance is achievable on the LAN without requiring a dynamic routing protocol.

TCP/IP supports ARP for VIPAs. In a flat network (one in which traffic flows directly between two endpoints without an intermediate router) using static routing with multiple interfaces onto the same LAN, you can achieve fault tolerance by defining a VIPA in the same subnet as the physical interfaces on the LAN. If a static route specifies a VIPA as the next hop IP address, the host or router will send an ARP for the VIPA. TCP/IP will reply to the ARP with the MAC address of one of the active physical interfaces on that LAN.

MPCOSA

The MPCOSA DEVICE statements define the MPC OSA Ethernet and FDDI devices.

MPCIPA

Use the MPCIPA DEVICE and LINK statements to define LAN connectivity via OSA-Express using the Queued Direct I/O (QDIO) interface. The MPCIPA device name must be the PORT name of the TRLE definition of the QDIO interface as described in *z/OS Communications Server: SNA Resource Definition Reference*. Device specifications for the type of IP routing supported are also specified on the MPCIPA DEVICE statement. These are also described in *z/OS Communications Server: SNA Resource Definition Reference*.

In configurations where multiple LCS and/or MPCIPA links onto the same LAN are defined, if the interface targeted by the ARP Request is inactive, one of the other active interfaces on the LAN will automatically take over responsibility for answering ARPs on behalf of the inactive interface. In this way, fault tolerance is achievable on the LAN without requiring a dynamic routing protocol.

TCP/IP supports ARP for VIPAs. In a flat network (one in which traffic flows directly between two endpoints without an intermediate router) using static routing with multiple interfaces onto the same LAN, you can achieve fault tolerance by defining a VIPA in the same subnet as the physical interfaces on the LAN. If a static route specifies a VIPA as the next hop IP address, the host or router will send an ARP for the VIPA. TCP/IP will reply to the ARP with the MAC address of one of the active physical interfaces on that LAN.

MPCPTP

MPCPTP can be used to define any of the following:

- A connection to another host over a series of CTCs (in this case, the device name must be the name of a VTAM TRLE).
- An XCF connection to another TCP/IP in the same z/OS sysplex. For an XCF connection, the device name must be the cp name of the target VTAM on the other side of the XCF connection, and the VTAM ISTLSXCF major node must be active to start the device.
- An IUTSAMEH connection (with no need for any I/O devices) to another TCP/IP on the same z/OS system or to VTAM for Enterprise Extender. For an IUTSAMEH connection, the device name must be the reserved name IUTSAMEH. VTAM automatically activates the IUTSAMEH TRLE.

Use the IPCONFIG DYNAMICXCF statement to cause TCP/IP to automatically define and activate XCF connectivity between each pair of TCP/IP stacks in the same sysplex and IUTSAMEH connectivity between multiple TCP/IP stacks on the same z/OS.

SNAIUCV and SNALU62

Use SNAIUCV DEVICE to specify the interface to use for SNA LU0 traffic to the SNALINK started procedures. For example, use this to define the interface between the TCP/IP address space and the SNALINK address space that is using a 3745 running NCPRoute. Similarly, the DEVICE SNALU62 statement defines the interface

between the TCP/IP address space and the address space using SNA LU6.2. Refer to *z/OS Communications Server: IP Configuration Reference* for information about how to define multiple LU6.2 connections within the same TCP/IP address space.

X.25 The DEVICE X25DEV defines the interface between the TCP/IP address space and the address space of the X.25 NPSI server.

VIPA and VIPADYNAMIC

Virtual IP Addresses (VIPA) are used to define virtual devices to the TCP/IP address space. There are two types of VIPAs:

- Static
- Dynamic

The static virtual device requires DEVICE and LINK statements to define a device that is always started, can never be stopped, can be known within the network, yet requires no physical adapters. It is very useful to define VIPAs so that if a physical adapter loses its connection to the network, application traffic using the failed physical adapter can be rerouted over another interface to the network. To the network, the VIPA address appears to be one hop away from the TCP/IP address spaces. The network sends and receives datagrams to and from the physical interfaces to get to the VIPA address. For more information about VIPA, see Chapter 5, "Virtual IP Addressing" on page 209.

INTERFACE

Use INTERFACE statements to define each IPv6 network interface to the TCPIP address space. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details about the various network interfaces supported by TCP/IP.

IPAQENET6

Use the IPAQENET6 INTERFACE statement to define IPv6 LAN connectivity through OSA-Express using QDIO.

VIRTUAL6

Use the VIRTUAL6 INTERFACE statement to define IPv6 static VIPAs.

HOME HOME lists the IP addresses and their associated LINK adapter. The first HOME statement within a configuration data set replaces the existing HOME list. If subsequent HOME statements are found within a configuration data set, add entries to the list.

Note: The order of the HOME list is important if IPCONFIG SOURCEVIPA is specified, except for TCP datagram requests with TCPSTACKSOURCEVIPA specified. The source address used will be the preceding VIPA address instead of the physical adapter used to send the datagram. If no VIPA precedes the physical adapter in the HOME list, the physical adapter IP address is used as the source address. Refer to *z/OS Communications Server: IP Configuration Reference* for precautions when either the VIPA address or a physical adapter used as a source for the VIPA has an IP address that is the network address.

PRIMARYINTERFACE

Use PRIMARYINTERFACE to specify which link should be designated as the default local host for use by the GETHOSTID() function. If

PRIMARYINTERFACE is not used, the first IP address in the HOME list becomes the default local host address.

BEGINROUTES
Use the BEGINROUTES statement to add static routes to the IP route table.

SMFCONFIG
Use the SMFCONFIG statement to provide SMF logging for Telnet, FTP, TCP, API, and stack activity. This statement is used for TYPE 118 and TYPE 119 records. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the SMFCONFIG statement.

START
Use START to activate a device or interface.

TRANSLATE
Use TRANSLATE to indicate which LINK has specified network addresses for use as a static ARP table. The first TRANSLATE statement in a configuration data set replaces the entire ARP cache. Subsequent TRANSLATE statements add to the table. If you are using OSPF routing (OMPROUTE), see Chapter 4, “Routing” on page 155 for more information about requirements for the TRANSLATE statement.

After an IPv4 interface has a DEVICE, LINK, and HOME statement, it can be started with the START device statement or the VARY TCPIP,,START command.

After an IPv6 interface has an INTERFACE statement, it can be started with the START interface statement or the VARY TCPIP,,START command.

Devices that support ARP offload

Certain devices provide an ARP offload function that offloads all ARP processing to the adapter. The function provided by the adapter impacts the ability of TCP/IP to display ARP cache information or ARP counter statistics for these devices.

Note: ARP processing is relevant only for IPv4 LAN interfaces.

The following devices provide an ARP offload function and provide ARP cache data or ARP counters to TCP/IP.

- MPCIPA (OSA-Express Gigabit Ethernet) with a minimum required microcode level of [MCL] 401
- MPCIPA (OSA-Express Fast Ethernet)
- MPCIPA (OSA-Express Token Ring)

Note: If multiple TCP/IP instances are sharing the device, the ARP data will represent all TCP/IP instances using the device. This information is provided to TCP/IP every 30 seconds from the device.

The following devices provide an ARP offload function and do not provide any ARP cache data or ARP counters to TCP/IP:

- MPCOSA (OSA-2 Fast Ethernet, FDDI)
- MPCIPA (OSA-Express Gigabit Ethernet) with a microcode level earlier than [MCL] 401

Note: For IPv6 LAN interfaces, TCP/IP performs all the neighbor discovery processing, maintains the neighbor cache, and provides the ability to display neighbor cache information.

HiperSockets concepts and connectivity

iQDIO (Internal Queued Direct Input/Output or HiperSockets) is a new S/390 zSeries hardware feature that provides high performance internal communications between LPARs within the same CEC without the use of any additional or external hardware equipment (for example, channel adapters, LANs, etc.). This support is also referred to as HiperSockets communications. When the processor supports HiperSockets and the CHPIDs have been configured in HCD (IOCP), TCP/IP connectivity can occur for two reasons:

- DYNAMICXCF is configured.
- A user defined iQDIO (MPCIPA) DEVICE and LINK is configured and started.

Therefore, there are two types of iQDIO devices:

- DYNAMICXCF iQDIO device (TRLE ″IUTIQDIO″ and an MPC group of subchannel devices). The PORTNAME will be IUTIQDxx, where xx = the IQD CHPID that VTAM uses (for example, IUTIQDFD when using IQD CHPID x'FD').
- A user defined iQDIO device (TRLE ″IUTIQDxx″ and an MPC group of subchannel devices). The PORTNAME is not applicable for this TRLE.

In both cases, the TRLE is dynamically built by VTAM. For additional details regarding how to configure a user defined iQDIO MPCIPA device refer to the *z/OS Communications Server: IP Configuration Reference*.

Concepts and considerations for the IQD CHPID: The iQDIO *hardware device* is represented by the IQD CHPID and its associated subchannel devices. All LPARs that are configured (HCD) to use the same IQD CHPID have internal connectivity and therefore have the capability to communicate using iQDIO. The IQD CHPID can be viewed as a *logical LAN* within the CEC. The iQDIO hardware allows up to 4 separate IQD CHPIDs to be defined per CEC, creating the capability of having 4 separate *logical LANs* within the same CEC. The following figures illustrate this concept:

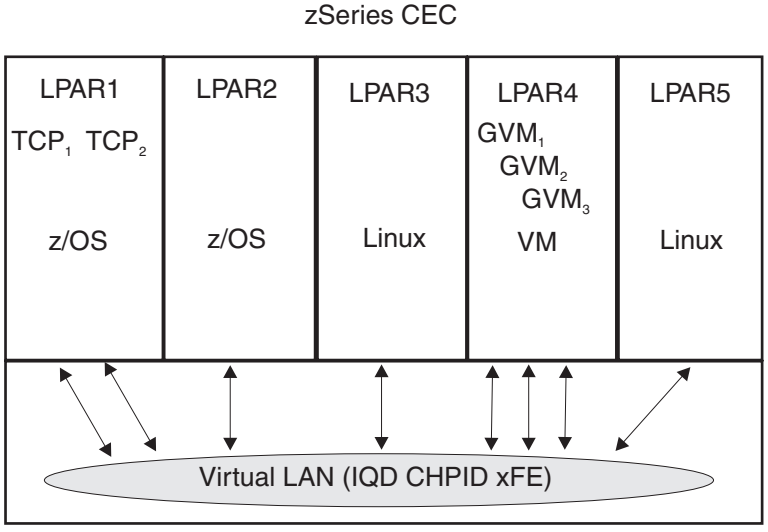


Figure 28. HiperSockets Virtual LAN

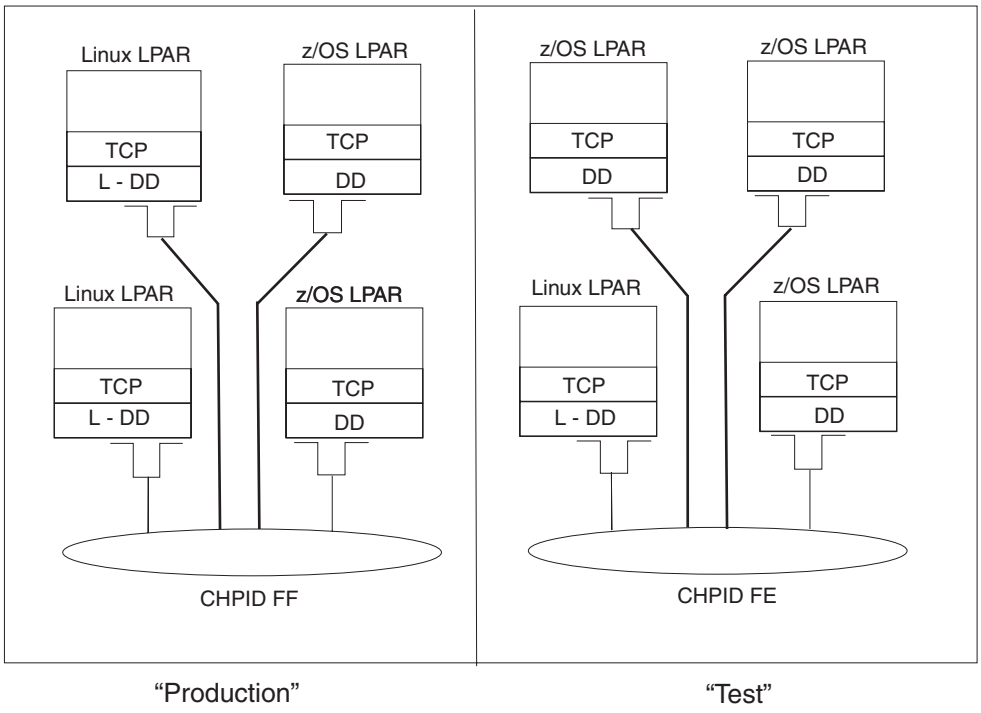


Figure 29. HiperSockets multiple LANs

Having this capability allows the system administrator to logically separate (or control) the internal connectivity, controlling which specific LPARs are allowed to internally connect using iQDIO. For example:

- SYSPLEX 'A' LPARs running on LPs 1 through 4 could use IQD CHPID x'FC'.
- SYSPLEX 'B' LPARs running on LPs 5 through 8 could use IQD CHPID x'FD'.
- A VM LPAR runs in LP 9 running various second level systems (Linux and z/OS) which use IQD CHPID x'FE'.
- combinations of the examples above could be:
  - Another set of LPARs on LPs 10 through 12 which are not using DYNAMICXCF (non SYSPLEX) are connected to IQD CHPIDs x'FE' and x'FF'.
  - Subsets of LPARs 1 through 8 are using both the DYNAMICXCF IQD CHPIDs and a non-DYNAMICXCF IQD CHPIDs.
  - Some LPARs are connected to all four IQD CHPIDs.

The iQDIO MPC group: VTAM will build a single iQDIO MPC group, using the subchannel devices associated with a single IQD CHPID. VTAM will use two subchannel devices for the read and write control devices, and 1 to 8 devices for data devices. Each TCP/IP stack will be assigned a single data device.

Therefore, in order to build the MPC group, there must be a minimum of 3 subchannel devices defined (within HCD) and associated with the same IQD CHPID. The maximum number of subchannel devices that VTAM will use is 10 (supporting 8 data devices or 8 TCP/IP stacks) per LPAR or MVS image. The subchannel devices must be configured for the LPAR and online prior to when the TCP/IP stack is initialized. Generally, the number of IQDIO subchannel devices you should configure per LPAR is:

```
2   (read / write control devices)
+ N  (where N = number of TCP/IP stacks)
-----------------
N+2  (total subchannel devices per LPAR)
Example (LPAR 1 starts two TCP/IP stacks and both stacks use iQDIO):

 -  define 4 subchannel devices on the same IQD CHPID
 -  where 2 are used for read / write control and 2 data devices are available
```

The first TCP/IP stack within the LPAR to initialize DYNAMICXCF will cause the iQDIO MPC group (IUTIQDIO) to be dynamically created. Each TCP/IP stack can then start the IUTIQDIO device, and each stack will be assigned a unique (dedicated) subchannel data device from the IUTIQDIO MPC group.

IBM recommends that the IQD CHPIDs be configured using CHPIDs x'FC' through x'FF' (but any valid CHPID value (x'00' through x'FF' can be configured as TYPE = IQD). Refer to *z/OS HCD Planning* and Appendix D, "Using HCD" on page 757 for additional details.

iQDIO Maximum Frame Size: The iQDIO hardware supports four different frame sizes referred to as the iQDIO MFS (Maximum Frame Size). Using HCD (or IOCP), the iQDIO MFS is configured on the IQD CHPID using the 'OS=' parameter. All LPARs communicating over the same IQD CHPID will then use the same IQD MFS. The MFS affects the largest packet that TCP/IP can transmit. TCP/IP will adjust the MTU (Maximum Transmission Unit) based on the MFS, which is discovered during activation.

The following table depicts the four possible TCP/IP MTU sizes resulting from the iQDIO frame sizes:

OS=*value*	iQDIO frame size	TCP/IP MTU size
00 (default)	16K	8K
40	24K	16K
80	40K	32K
C0	64K	56K

The default iQDIO MFS is 16K. However, in cases in which increased bandwidth is required (such as large file transfers, file backup, etc.), a larger MFS could be used. In most workload environments the default size will result in better storage and CPU utilization.

```
************************************************************
* OS  values are '00'=16K, '40'=24K, '80'=40K  and 'C0'=64K.  *
*                                                          *
* Need at least 3 addresses per z/OS, maximum of 10:       *
*   - 2 addresses for control                              *
*   - 1 address for data for each TCP stack (between 1 and 8) *
************************************************************
 ID SYSTEM=(2064,1)
*
 CHPID PATH=FC,TYPE=IQD,SHARED,OS=00
 CHPID PATH=FD,TYPE=IQD,SHARED,OS=40
 CHPID PATH=FE,TYPE=IQD,SHARED,OS=80
 CHPID PATH=FF,TYPE=IQD,SHARED,OS=C0
*
 CNTLUNIT  CUNUMBR=FC00,PATH=FC,UNIT=IQD
 IODEVICE  ADDRESS=(2C00,16),CUNUMBR=FC00,UNIT=IQD
*
 CNTLUNIT  CUNUMBR=FD00,PATH=FD,UNIT=IQD
```

```
 IODEVICE  ADDRESS=(2C10,16),CUNUMBR=FD00,UNIT=IQD
*
 CNTLUNIT  CUNUMBR=FE00,PATH=FE,UNIT=IQD
 IODEVICE  ADDRESS=(2C20,16),CUNUMBR=FE00,UNIT=IQD
*
 CNTLUNIT  CUNUMBR=FF00,PATH=FF,UNIT=IQD
 IODEVICE  ADDRESS=(2C30,16),CUNUMBR=FF00,UNIT=IQD
```

Refer to *z/OS HCD Planning* and Appendix D, “Using HCD” on page 757 for additional details.

Modifying iQDIO connectivity (TCP/IP device and link and the VTAM iQDIO (IUTIQDIO) MPC group): Certain modifications can be made to the iQDIO device (MPC group) without disrupting an active TCP/IP stack.

z/OS supports dynamic I/O for the iQDIO CHPID and subchannel devices allowing subchannels devices to be added or removed to or from an LPAR which has already been IPLed.

TCP/IP supports the STOP and START command for the iQDIO (IUTIQDIO) device. However, the commands are only supported when the (internal) start (activation) was successful during stack initialization. TCP/IP also supports the STOP and START command for the user defined iQDIO devices (IUTIQDxx). Since a user defined iQDIO device is supported as an MPCIPA device, STOP and START function just as they would for other MPCIPA devices.

VTAM supports a MODIFY IQDCHPID command, which allows the user to change the initial setting of the IQDCHPID start option.

Therefore, it is possible to make certain changes to the DYNAMICXCF iQDIO MPC Group (IUTIQDIO) without restarting VTAM or an active TCP/IP stack. Examples of changes that can be made are (STOP/START device required):

- Alter which specific IQD CHPID is used for DYNAMICXCF (for example, move from the x’FC’ CHPID to the x’FD’ CHPID).
- Add or remove subchannel devices (for example, from the current IQD CHPID).
- Alter the IQD MFS which alters the TCP/IP MTU (for example, increase the current IQD CHPID from 16k to 64k).

Although VTAM supports modifications to the start option IQDCHPID (and the modification will be immediately displayed), the effects will vary depending on what the current usage was and the change (from or to) that was made. For example:

- When MODIFIED from ANY (or CHPID) to NONE, there no effect on current usage but blocks subsequent activations of the DYNAMICXCF iQDIO device
- When MODIFIED from NONE to ANY (or CHPID), there is no effect on current usage but allows subsequent activations.
- When MODIFIED from CHPID_X to CHPID_Y, there is no effect on current usage.

  Note: VTAM only uses the CHPID value when building the IUTIQDIO MPC group.

  To change CHPIDs for an active MPC group the following must be done:

  1. TCP/IP IUTIQDIO devices that are changing must be stopped.
  2. Make any necessary HCD/IOCDS changes.
  3. Verify new subchannel devices are varied online.

4. Verify the MPC group has deactivated (with no usage it times out after approximately 2 minutes).
5. Modify IQDCHPID = CHPID (to new CHPID).
6. Restart the TCP/IP IUTIQDIO devices.

In order to use iQDIO communications, the processor must have the necessary hardware support. If the processor does not support iQDIO communications, modifications to this start option will not be accepted, and the IQDCHPID option will not be displayed (displayed as ***NA***) .

iQDIO connectivity and routing: For each pair of stacks within a sysplex (which are not on the same MVS image), if all of the following conditions are true, then the stacks will use iQDIO DYNAMICXCF connectivity (versus standard XCF links):

- The two stacks must be on the same CEC
- The two stacks must be using the same IQD CHPID for the DYNAMICXCF iQDIO (IUTIQDIO) device
- Both stacks must be at the z/OS V1R2 (or higher) level and be configured (HCD) to use iQDIO
- The initial iQDIO activation must complete successfully.

If any of the above conditions are not met, then the stacks will use XCF connectivity.

When a DYNAMICXCF iQDIO device and link are created and successfully activated, a subnet route is created across the iQDIO link. The subnet is created by using the DYNAMICXCF IP address and mask. This allows any LPAR within the same CEC to be reached, even ones that are not within the sysplex. For example, an LPAR that is running Linux and does not support joining the sysplex can still be reached. The Linux LPAR must define at least one IP address for the iQDIO endpoint that is within the subnet defined by the DYNAMICXCF IP address and mask.

Therefore, TCP/IP can communicate with other LPARs within the CEC over the DYNAMICXCF iQDIO (IUTIQDIO) device even when the TCP/IP in the other LPAR is not part (joins or supports) of the sysplex. You can also elect to manually configure an iQDIO device for non-sysplex communications.

When multiple stacks reside within the same LPAR which supports iQDIO, both IUTSAMEH and iQDIO links will coexist. In this case, it is possible to transfer data across either link. Because IUTSAMEH links have better performance, it is better to always use them for intra-stack communication. A host route will be created by DYNAMICXCF processing across the IUTSAMEH link but not across the iQDIO link. To avoid using the iQDIO link for communication within the same host, the following rules should be observed:

- Specify DYNAMICXCF IP addresses within a separate subnet from VIPA addresses.
- Do not specify static IUTSAMEH links.

It is also possible with multiple stacks in the same LPAR to end up with both XCF and iQDIO links. This occurs when the availability of the (preferable) iQDIO link changes as each TCP stack (within the same LPAR) is started. For example, stack A is started with iQDIO available and later stack B is started with iQDIO unavailable. This type of configuration should be avoided.

Efficient routing using HiperSockets Accelerator: Communications Server leverages the technological advances and high performing nature of the I/O processing offered by HiperSockets with the IBM zSeries servers and the IBM OSA-Express using QDIO architecture by optimizing IP packet forwarding processing that occurs across these two types of links. This function is referred to as HiperSockets Accelerator. It is a configurable option, and activated by configuring the IQDIORouting option on the IPCONFIG statement.

When configured, it allows IP packets that are forwarded across an iQDIO link from a QDIO link (or from QDIO to iQDIO) to be forwarded by the z/OS CS HiperSockets device driver. That is, the IP forwarding function is *pushed down* as close to the hardware [or to the lowest software DLC (Data Link Control)] layer as possible so that these packets do not have to be processed by the TCP/IP stack or address space. Therefore, valuable TCP/IP resources (storage and machine cycles) are not expended for purposes of routing and forwarding packets. The following figure illustrates a configuration before the utilization of HiperSockets Accelerator.

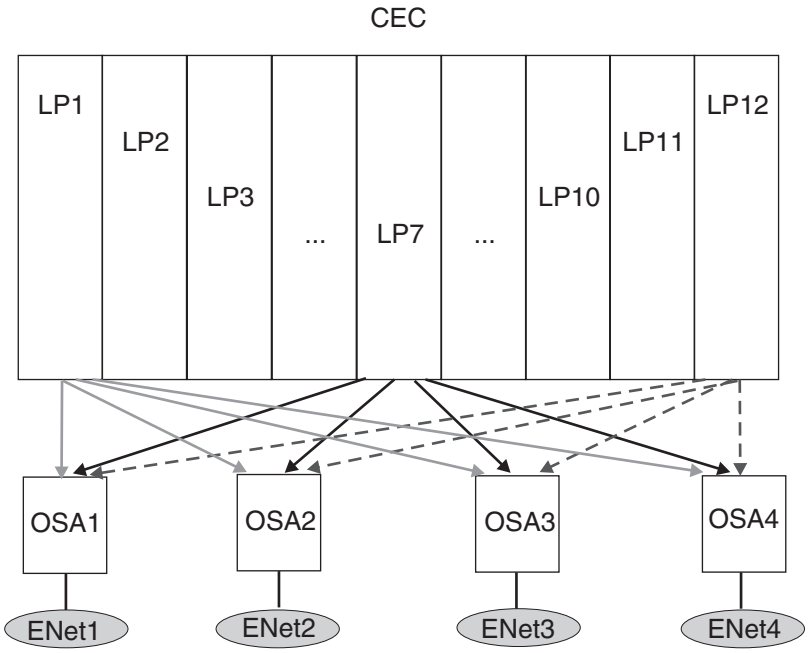


Figure 30. Candidate configuration for HiperSockets Accelerator

HiperSockets Accelerator presents a different configuration and approach to obtain full connectivity as shown in the figure below.

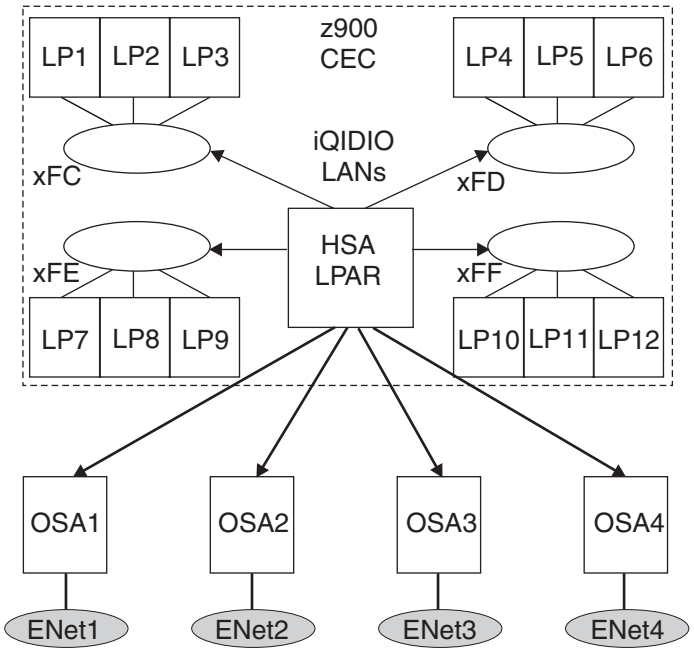


Figure 31. HiperSockets Accelerator configuration

This function allows a user to position a specific or single TCP/IP stack which has direct physical connectivity to the OSAs LANs as the iQDIO router. This stack can then connect to all remaining TCP/IP stacks in other images (LPARs) within the same CEC that require connectivity to the same OSA LANs using HiperSockets connectivity.

This approach becomes more beneficial as the number of LPARs within a given CEC increase. Instead of attempting to directly attach each LPAR to each physical network attachment using an OSA LAN, a smaller number of OSAs could be concentrated through a single z/OS LPAR. From a performance perspective, HiperSockets Accelerator attempts to make the intermediate (or router) TCP/IP stack appear as if it did not exist in the path. Instead, each LPAR will appear as if each were directly attached to the physical network (for example, packets are forwarded without traversing the router TCP/IP stack). There are no additional routing configuration tasks required by the user. The prerouting occurs automatically. The TCP/IP stack automatically detects IP packet forwarding is occurring across a HiperSockets eligible route (QDIO/iQDIO or iQDIO/QDIO), and dynamically creates an IDIORouting route entry. All subsequent packets will then take the optimized device driver path, and will not traverse the TCP/IP stack.

The dynamically created iQDIO routing entries can be displayed with NETSTAT. VTAM tuning statistics are provided to allow the user to monitor or measure prerouting activity.

IQDIOPriority (IQDIORouting option) is an optional choice that allows the user to specify which of the four priority queues should be used when prerouting packets from an iQDIO link outbound to a QDIO link. The default is 1 (highest priority), and in most cases should be sufficient.

For additional details regarding the IQDIORouting configuration option, refer to the IPCONFIG statement in the *z/OS Communications Server: IP Configuration Reference*.

Interface-layer fault-tolerance for local area networks (interface-takeover function)

The TCP/IP stack in the z/OS Communications Server provides transparent fault-tolerance for failed (or stopped) IPv4 devices or IPv6 interfaces, when the stack is configured with redundant connectivity onto a LAN. This support is provided by the z/OS Communications Server interface-takeover function, and applies to IPv4 MPCIPA and LCS device types and to the IPv6 IPAQENET6 interface type.

At device or interface startup time, TCP/IP dynamically learns of redundant connectivity onto the LAN, and uses this information to select suitable backups in the case of a future failure of the device or interface. This support makes use of ARP flows (for IPv4 devices) or neighbor discovery flows (for IPv6 interfaces), so upon failure (or stop) of a device or interface, TCP/IP immediately notifies stations on the LAN that the original IPv4 or IPv6 address is now reachable through the backup's link-layer (MAC) address. Users targeting the original IP address will see no outage due to the failure, and will be unaware that any failure occurred.

Since this support is built upon ARP or neighbor discovery flows, no dynamic routing protocol in the IP layer is required to achieve this fault tolerance. To enable this support, you only need to configure redundancy onto the LAN:

- You need redundant LAN adapters.
- For IPv4, you must configure and activate multiple LINKs onto the LAN.
- For IPv6, you need to configure and start multiple INTERFACEs onto the LAN.

Note: An IPv4 device cannot back up an IPv6 interface, and an IPv6 interface cannot back up an IPv4 device.

The interface-layer fault-tolerance feature can be used in conjunction with VIPA addresses, where applications can target the VIPA address, and any failure of the real LAN hardware is handled by the interface-takeover function. This differs from traditional VIPA usage, where dynamic routing protocols are required to route around real hardware failures.

IPv6 considerations: Stateless autoconfiguration and duplicate address detection

IPv6 provides the capability of autoconfiguring addresses for an interface by using information provided by IPv6 routers. Descriptions of this function can be found in RFC 2461 and RFC 2462. The term *autoconfigured IP address* is used below to mean an IP address that is created as a result of information received from a router advertisement. z/OS TCP/IP allows autoconfiguration if no IP addresses are defined on the profile INTERFACE statement using the IPADDR keyword. If the INTERFACE statement contains IPADDR definitions, this indicates that the installation is defining its own IP addresses and autoconfiguration is not desired. Subsequent descriptions use the term *manually configured addresses* to describe the addresses that are defined using the IPADDR keyword.

TCP creates an autoconfigured IP address for an interface if all three of the following conditions are met:

- The interface is active.
- A valid router advertisement containing prefix information with the autonomous flag on is received over the interface.
- No manually configured home addresses are defined for the interface at the time the router advertisement is received.

The IP address that is created is formed by appending the interface ID generated by the stack to the prefix supplied by the router advertisement. Autoconfigured IP addresses can be identified in the netstat home report by the 'Autoconfigured' flag. For more information on the interface ID generated by the stack, see *z/OS Communications Server: IPv6 Network and Application Design Guide*.

An autoconfigured IP address exists until one of the following occurs:

- The valid lifetime specified by the most recent router advertisement expires. When the valid lifetime expires, the autoconfigured address is removed. Existing connections using this address are terminated when subsequent activity occurs on the connection. The router advertisement that contains the valid lifetime for the autoconfigured address can also specify a preferred lifetime. The preferred lifetime indicates that the IP address can be freely used. When the preferred lifetime expires, the autoconfigured address is considered deprecated. The deprecated state indicates that another IP address should be used if available and provides a transition period before the valid lifetime expires. A deprecated IP address can be identified in the netstat home report by the 'deprecated' flag.
- The installation activates a profile that contains a manually configured IP address on the same interface as the autoconfigured IP address (that is, the INTERFACE statement contains the ADDADDR keyword). If this occurs, any autoconfigured IP addresses on that interface are deleted and existing connections using this address are terminated when subsequent activity occurs on the connection. The manually configured addresses are added and duplicate address detection for the newly added IP addresses initiated, if applicable.

Duplicate address detection is the process described in RFC 2462 which verifies that IPv6 home addresses are unique on the local link before assigning them to an interface. Duplicate address detection is performed on all IPv6 IPAQENET6 home addresses, whether they are manually configured or autogenerated, unless the INTERFACE statement specifies DUPADDRDET 0. Duplicate address detection is not done for LOOPBACK6 or VIRTUAL6 addresses. The duplicate address detection process sends a multicast neighbor solicitation and waits a period of time to see if another neighbor indicates that the address is in use. By default, only one neighbor solicitation is sent and the length of time waited is approximately one second. If no neighbor responds in that interval, the address is considered unique and the interface will start using it. The number of neighbor solicitations sent by duplicate address detection can be modified by the DUPADDRDET keyword on the INTERFACE statement. The duration of the wait interval (awaiting a response from a neighbor already using the address) can be modified by information obtained from routers on the attached network.

Duplicate address detection occurs when the interface is started. Unless the INTERFACE statement indicated duplicate address detection is to be bypassed, IPv6 manually configured addresses are unavailable until the interface is started and duplicate address detection completes without finding another node on the local link with the same address. Prior to activation of the interface, manually configured addresses are shown in the netstat home report as unavailable with a reason of 'DUPLICATE ADDRESS DETECTION PENDING'. While the duplicated address detection is actively in progress for an address, the netstat home report shows the address as unavailable with a reason of 'DUPLICATE ADDRESS DETECTION IN PROGRESS'. If another neighbor indicates the address is in use during the duplicate address detection process, message EZZ9780I is issued and the address is not made available to the interface. If the address that failed duplicate address

detection is a manually configured address, the address shows up in the netstat home report as unavailable with a reason of 'DUPLICATE ADDRESS DETECTION FAILED'.

A link-local address is required to activate a QDIO IPv6 interface and will be generated automatically by the stack. The link-local address is generated using the link-local prefix and the interface ID. If the link-local address generated from the interface ID is determined to be a duplicate, the interface is not activated if:

- Autoconfigured addresses are allowed.
- A manually configured home address specifying only the prefix was specified on the INTERFACE statement. In such a case, the interface ID needs to be used to form the complete link-local address, and since the interface ID has been found to be in use on the network, the formed link-local address cannot be used.

If duplicate address detection fails on the link-local address and only fully configured manual addresses were specified on the INTERFACE statement, up to two attempts are made to create a unique link local address using a randomly generated value instead of the interface ID. If duplicate address detection succeeds using the randomly generated link-local address, message EZZ9784I is issued indicating the generated address and the interface is activated.

Setting up reserved port number definitions in PROFILE.TCPIP

Figure 32 shows a portion of the sample configuration file for the TCP/IP address space, PROFILE.TCPIP. This sample can be copied from SEZAINST(SAMPPROF). Figure 32 includes the portion of the sample that shows how to set up reserved port number definitions. Descriptions for the statements follow Figure 32. For more information about any of these statements, refer to *z/OS Communications Server: IP Configuration Reference*. For information specific to IPv6 support, refer to *z/OS Communications Server: IPv6 Network and Application Design Guide*.

Figure 32. Example of reserved port number definitions

```
; ====================================================================
; Application configuration
; ====================================================================
;
;
; AUTOLOG: Supplies TCPIP with the procedure names to start and the
; time value to wait at TCP start up for any of those procedures
; to terminate if they are active.
;
; AUTOLOG 5
;  FTPD JOBNAME FTPD1       ; FTP Server
;  LPSERVE                  ; LPD Server
;  NAMED                    ; Domain Name Server
;  NCPROUT                  ; NCPROUTE Server
;  OROUTED                  ; OROUTED Server
;  OSNMPD                   ; SNMP Agent Server
;  PORTMAP                  ; Portmap Server (SUN 3.9)
;  PORTMAP JOBNAME PORTMAP1 ; USS Portmap Server (SUN 4.0)
;  RXSERVE                  ; Remote Execution Server
;  SMTP                     ; SMTP Server
;  SNMPQE                   ; SNMP Client
;  TCPIPX25                 ; X25 Server
; ENDAUTOLOG
;
;
; PORT: Reserves a port for specified job names
;
;   - A port that is not reserved in this list can be used by any user.
;     If you have TCP/IP hosts in your network that reserve ports
;     in the range 1-1023 for privileged applications, you should
;     reserve them here to prevent users from using them.
```

```
;     The RESTRICTLOWPORTS option on TCPCONFIG and UDPCONFIG will also
;     prevent unauthorized applications from accessing unreserved
;     ports in the 1-1023 range.
;
;   - A PORT statement with the optional keyword SAF followed by a
;     1-8 character name can be used to reserve a PORT and control
;     access to the PORT with a security product such as RACF.
;     For port access control, the full resource name for the security
;     product authorization check is constructed as follows:
;     EZB.PORTACCESS.sysname.tcpname.safname
;     where:
;       EZB.PORTACCESS is a constant
;       sysname is the MVS system name (substitute your sysname)
;       tcpname is the TCPIP jobname (substitute your jobname)
;       safname is the 1-8 character name following the SAF keyword
;
;     When PORT access control is used, the TCP/IP application
;     requiring access to the reserved PORT must be running under a
;     USERID that is authorized to the resource. The resources
;     are defined in the SERVAUTH class.
;
;     For an example of how the SAF keyword can be used to enhance
;     security, see the definition below for the FTP data PORT 20
;     with the SAF keyword. This definition reserves TCP PORT 20 for
;     any jobname (the *) but requires that the FTP user be permitted
;     by the security product to the resource:
;     EZB.PORTACCESS.sysname.tcpname.FTPDATA in the SERVAUTH class.
;
;   - The BIND keyword is used to force a generic server (one that
;     binds to INADDR_ANY) to bind to the specific IP address that
;     is specified following the BIND keyword. This capability could
;     be used, for example, to allow z/OS UNIX telnet and telnet
;     3270 servers to both bind to TCP port 23.
;     The IP address that follows bind must be in IPv4 dotted
;     decimal format and may be any valid address for the host
;     including VIPA and dynamic VIPA addresses.
;
;   The special jobname of OMVS indicates that the PORT is reserved
;   for any application with the exception of those that use the Pascal
;   API.
;
;   The special jobname of * indicates that the PORT is reserved
;   for any application, including Pascal API socket applications.
;
;   The special jobname of RESERVED indicates that the PORT is
;   blocked. It will not be available to any application.
;
;   The special jobname of INTCLIEN indicates that the PORT is
;   reserved for internal stack use.
;
;
PORT
     7 UDP MISCSERV            ; Miscellaneous Server - echo
     7 TCP MISCSERV            ; Miscellaneous Server - echo
     9 UDP MISCSERV            ; Miscellaneous Server - discard
     9 TCP MISCSERV            ; Miscellaneous Server - discard
    19 UDP MISCSERV            ; Miscellaneous Server - chargen
    19 TCP MISCSERV            ; Miscellaneous Server - chargen
    20 TCP *  NOAUTOLOG        ; FTP Server
;   20 TCP *  NOAUTOLOG SAF FTPDATA ; FTP Server
    21 TCP FTPD1               ; FTP Server
;   21 TCP FTPD2 BIND FEC9:C2D4:1:0000:0009:0067:0115:0066 ; FTP IPv6
    23 TCP INTCLIEN            ; Telnet 3270 Server
;   23 TCP INETD1 BIND 9.67.113.3 ; z/OS UNIX Telnet server
    25 TCP SMTP                ; SMTP Server
    53 TCP NAMED               ; Domain Name Server
    53 UDP NAMED               ; Domain Name Server
   111 TCP PORTMAP             ; Portmap Server (SUN 3.9)
   111 UDP PORTMAP             ; Portmap Server (SUN 3.9)
;  111 TCP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
;  111 UDP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
   123 UDP SNTPD               ; Simple Network Time Protocol Server
   135 UDP LLBD                ; NCS Location Broker
   161 UDP OSNMPD              ; SNMP Agent
   162 UDP SNMPQE              ; SNMP Query Engine
   389 TCP LDAPSRV             ; LDAP Server
```

```
   443 TCP HTTPS               ; http protocol over TLS/SSL
   443 UDP HTTPS               ; http protocol over TLS/SSL
   512 TCP RXSERVE             ; Remote Execution Server
   514 TCP RXSERVE             ; Remote Execution Server
;  512 TCP * SAF OREXECD       ; z/OS UNIX Remote Execution Server
;  514 TCP * SAF ORSHELLD      ; z/OS UNIX Remote Shell Server
   515 TCP LPSERVE             ; LPD Server
   520 UDP OROUTED             ; OROUTED Server
   580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERB             ; Kerberos
   750 UDP MVSKERB             ; Kerberos
   751 TCP ADM@SRV             ; Kerberos Admin Server
   751 UDP ADM@SRV             ; Kerberos Admin Server
  1933 TCP ILMTSRVR            ; IBM LM MT Agent
  1934 TCP ILMTSRVR            ; IBM LM Appl Agent
  3000 TCP CICSTCP             ; CICS Socket
  3389 TCP MSYSLDAP            ; LDAP Server for Msys
;
;
; PORTRANGE: Reserves a range of ports for specified jobnames.
;
;   In a common INET (CINET) environment, the port range indicated by
;   the INADDRANYPORT and INADDRANYCOUNT in your BPXPRMxx parmlib member
;   should be reserved for OMVS.
;
;   The special jobname of OMVS indicates that the PORTRANGE is reserved
;   for ANY z/OS UNIX socket application.
;
;   The special jobname of * indicates that the PORTRANGE is reserved
;   for any socket application, including Pascal API socket
;   applications.
;
;   The special jobname of RESERVED indicates that the PORTRANGE is
;   blocked. It will not be available to any application.
;
;   The SAF keyword is used to restrict access to the PORTRANGE to
;   authorized users. See the use of SAF on the PORT statement above.
;
;
;   PORTRANGE 4000 1000 TCP OMVS
;   PORTRANGE 4000 1000 UDP OMVS
;   PORTRANGE 2000 3000 TCP RESERVED
;   PORTRANGE 5000 6000 TCP * SAF RANGE1
;
; SACONFIG: Configures the TCP/IP SNMP subagent
;
SACONFIG ENABLED COMMUNITY public AGENT 161
```

The following explains the statements shown in Figure 32 on page 139.

AUTOLOG

Use AUTOLOG to list the procedure names that should start when the TCPIP address space starts. It is also used to supply a timeout value for detecting hung procedures at TCP/IP initialization time. The timeout value is the time TCP/IP should allow for a procedure to come down when, at startup, it is still active and TCP/IP is attempting to AUTOLOG the procedure again. A hung procedure is active to MVS, but is not listening on the socket that is reserved for it via the PORT statement. When AUTOLOG detects a hung task, TCP/IP checks every 10 seconds (until the timeout value has expired) to see if the procedure has come down. If the procedure comes down during one of these 10 second intervals, it is restarted. If the procedure is still active when the time interval specified by the timeout value expires, then TCP/IP cancels and restarts the procedure.

The AUTOLOG statement shown in Figure 32 on page 139 has a timeout value of five minutes.

In the first AUTOLOG statement the FTP Server shows FTPD JOBNAME FTPD1. This means when the TCPIP address space starts, the FTPD procedure will be started via the MVS START FTPD command. Because

FTPD forks a child process that actually listens on PORT 21 (see the PORT statement in this section), the autolog task verifies that FTPD1 is listening on port 21.

Similarly, when the TCPIP address space starts, the autolog task starts the remaining 10 tasks.

Unless the tasks in the AUTOLOG list are in the PORT reservation list, the autolog task does not check for hung tasks every five minutes. Also at startup time, those procedures that are not on the PORT list are first canceled and then started. This occurs because the procedure might have been running from a previous TCP/IP address space and would need to be started and stopped to start listening when the new stack starts.

Notes:

1. If you run multiple TCP/IP address spaces, ensure that the second address space AUTOLOG list does not cancel the procedures of the first. In those cases, an installation might require different procedure names for the servers for each address space. For more information about multiple stacks, see "Port management overview" on page 55.
2. You can use the AUTOLOG statement to automatically start generic servers in a single stack environment, but you should be careful using the AUTOLOG statement to start generic servers in a multiple stack environment. Instead, you could use an operations automation software package (IBM and other vendors provide these) to start generic servers automatically. For a list of generic servers provided by TCP/IP, see "Generic servers in a CINET environment" on page 58.

For those procedures that require parameters to be used on the MVS START command, there is a PARMSTRING option. For more information, refer to *z/OS Communications Server: IP Configuration Reference*.

PORT Use PORT to reserve ports for different jobs. This prevents a rogue application from taking port 21, which is needed by FTP. For each port entry, the port number, protocol, and procedure name are specified. The first port entry shows port 7 UDP reserved for the miscellaneous echo server for procedure MISCSERV. Similarly, port 7 of TCP is also reserved for the same server. In this example, six ports are reserved for the miscellaneous server.

INTCLIEN is an INTernal CLIENt to the TCPIP address space (the TN3270 Telnet server), and it runs continuously. See Chapter 8, "Accessing remote hosts using Telnet" on page 305 for more information about INTCLIEN.

NOAUTOLOG can be specified, as in the port 20 TCP * in Figure 32 on page 139. In this way, the port is reserved for an OMVS forked task so that the FTP server can fork tasks to port 20 as each FTP user logs in.

Use the DELAYACKS and NODELAYACKS options to allow an installation to delay their acknowledgments so they can be combined with data to be sent to foreign hosts. Unless a performance reason is needed, NODELAYACKS should be used to immediately send acknowledgments.

Use SHAREPORT when reserving a port to be shared across multiple TCP listeners. This is not valid for UDP.

Typically, reserving a port for a specific job name is sufficient. If the port must instead be reserved for a specific user ID or a set of user IDs, use the SAF keyword to specify the name of a SAF resource to be associated with

the port. The user ID associated with the application that attempts to bind to the port must be permitted to the SAF resource.

The BIND keyword is used to force a generic server (one that binds to INADDR_ANY) to bind to the specific IP address that is specified following the BIND keyword. This capability could be used, for example, to allow the z/OS UNIX Telnet and TN3270 Telnet servers to both bind to TCP port 23 on different IP addresses. The IP address that follows bind can be any valid address for the host, including VIPA and dynamic VIPA addresses. The address supplied can be either an IPv4 address (in dotted-decimal format) or an IPv6 address (in colon-hexadecimal format). IPv4-mapped IPv6 addresses and IPv4-compatible IPv6 addresses are not supported. For multiple servers to bind to the same port with this function, the IP address for each server must be unique.

RESERVED indicates that the port is not available for use by any user.

PORTRANGE
PORTRANGE is a statement used to reserve a range of ports for specified job names.

SACONFIG
SACONFIG is the statement used to configure the information about the SNMP subagent. Omission of this statement causes TCPIP to assume the default value of SACONFIG ENABLED COMMUNITY public AGENT 161. Use SACONFIG to specify the following:

- SNMP community string
- OSA/SF port number
- Agent port
- OSA management support
- Whether or not the SNMP agent can perform SNMP sets

Setting up SAF Server Access Authorization (SERVAUTH) (optional)

The TCP/IP address space uses the SERVAUTH System Authorization Facility (SAF) class to protect TCP/IP resources from unauthorized access. The use of SERVAUTH may be optional and is available in degrees so that installations can pick and choose the access needed. Installations may be able to choose to use one, all, or none of the protections provided by SERVAUTH. The customizing described in this section is completely optional when using the IBM security product RACF. Non-IBM security products may require customizing. A template of the commands and all other SAF commands appears in SEZAINST(EZARACF). Refer to Chapter 2, “Security” on page 79 for more detailed information.

Configuring the local host table (optional)

Why configure a local host table?

You can set up the local host table to support local host name resolution. If you use the local host table for this purpose, your socket applications will only be able to resolve names and IP addresses that appear in your local host table.

If you need to resolve host names outside your local area, you can configure the resolver to use a domain name server (see the NSINTERADDR statement). If you use a domain name server, you do not need to set up any host definitions in your resolver configuration, but you may still do so.

If you have configured your resolver to use a name server, it will always try to do so, unless your TCP/IP C/C++ API applications were written with a RESOLVE_VIA_LOOKUP symbol in the source code. You can also configure the resolver to only use a local host table by specifying LOOKUP LOCAL in the TCPIP.DATA configuration file. For both cases, all name resolution calls will always use a local host table. This is probably not a technique you will see for standard socket applications, but it may be a technique you could find useful for when you develop your own socket applications or for testing changes before they are placed in your name server.

It might be a good idea to have a local host table available for the resolver to use if the name server is not reachable. If the name server does not respond to name resolution requests, the resolver tries to use the local host table. If the name server is reachable but returns a negative reply for a name resolution request, the resolver tries to resolve the name using the local host table, if such a file is present.

Assume you try to resolve the host name *friendly* and your DOMAINORIGIN is *my.house.com*, the resolver sends a query to the name server for *friendly.my.house.com*. If the name server returns a negative reply (the name is not registered), the resolver looks into the local host table for an entry of *friendly.my.house.com* and, if not found, for an entry of *friendly*.

Due to the flexibility of the Domain Name System, it is recommended you use a domain name server. If you set up a small TCP/IP network, the simplicity of the local host table approach might be preferable.

The following types of local host table can be used:

- HOSTS.LOCAL

  HOSTS.LOCAL is only used for IPv4 requests.
- /etc/hosts

  /etc/hosts is only used for IPv4 requests.
- ETC.IPNODES and /etc/ipnodes

  ETC.IPNODES and /etc/ipnodes can be used for IPv6 requests, and for IPv4 requests when COMMONSEARCH is coded in the resolver setup statements.

Creating HOSTS.LOCAL site host table

The site host table is generated from the *hlq*.HOSTS.LOCAL data set. This data set contains descriptions of local host entries in the HOSTS format. HOSTS.LOCAL can only contain IPv4 addresses. A sample HOSTS.LOCAL data set is created during installation. The following sections describe how to update the sample *hlq*.HOSTS.LOCAL data set and use it to generate the two data sets, *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO, which function as your site table.

HOST entries

One line of the *hlq*.HOSTS.LOCAL data set is used for each distinct host and ends with four colons (::::). The maximum length of the line is 512 characters. Each host can have multiple IP addresses and multiple names. The line for each host has three essential fields, separated by colons. These fields are:

- The keyword *HOST*
- A list, separated by commas, of IP addresses for that host. A maximum of 6 IP addresses can be specified.

- A list, separated by commas, of fully qualified names for that host. A maximum of 20 host names can be specified. Only the first six host names will be used in the *hlq*.HOSTS.ADDRINFO data set. All twenty host names will be used in the *hlq*.HOSTS.SITEINFO data set.

For example, if you have two local hosts, LOCAL1 (IP addresses 192.6.77.4 and 192.8.4.1) and LOCAL2 (with an alias LOCALB and IP address 192.6.77.2), append the following lines to the *hlq*.HOSTS.LOCAL data set:

```
HOST : 192.6.77.4, 192.8.4.1 : LOCAL1 ::::
HOST : 192.6.77.2 : LOCAL2, LOCALB ::::
```

Note: The maximum length for a host allowed in the HOST tables is 24 characters.

NET and GATEWAY entries

The NET and GATEWAY statements are not used by TCP/IP for z/OS applications. However, some socket calls require the NET entries. If your programs do not need the NET and GATEWAY statements, delete them before invoking MAKESITE.

Sample HOSTS.LOCAL data set (HOSTS): Following is the sample HOSTS.LOCAL data set provided in SEZAINST(HOSTS):

```
; HOSTS.LOCAL
; -----------
; COPYRIGHT = NONE.
;
; The format of this file is documented in RFC 952, "DoD Internet
; Host Table Specification".
;
; The format for entries is:
;
; NET : ADDR : NETNAME :
; GATEWAY : ADDR, ALT-ADDR : HOSTNAME : CPUTYPE : OPSYS : PROTOCOLS :
; HOST : ADDR, ALT-ADDR : HOSTNAME, NICKNAME : CPUTYPE : OPSYS : PROTOCOLS :
;
; Where:
;   ADDR, ALT-ADDR = IP address in decimal, e.g., 26.0.0.73
;   HOSTNAME, NICKNAME = the fully qualified host name and any nicknames
;   CPUTYPE = machine type (PDP-11/70, VAX-11/780, IBM-3090, C/30, etc.)
;   OPSYS = operating system (UNIX, TOPS20, TENEX, VM/SP, etc.)
;   PROTOCOLS = transport/service (TCP/TELNET,TCP/FTP, etc.)
;   : (colon) = field delimiter
;   :: (2 colons) = null field
; *** CPUTYPE, OPSYS, and PROTOCOLS are optional fields.
;
;   MAKESITE does not allow continuation lines, as described in
;   note 2 of the section "GRAMMATICAL HOST TABLE SPECIFICATION"
;   in RFC 952.  Entries should be specified on a single line of
;   up to a maximum of 512 characters per line.
;
;
; Note:  The NET and GATEWAY statements are not used by the TCP/IP for
;        MVS applications.  However, some socket calls require the NET
;        entries.  For better performance, if your programs do not need
;        the NET and GATEWAY statements, delete them before running
;        the MAKESITE program.
;
;
HOST : 9.67.43.100 : NAMESERVER ::::
HOST : 9.67.43.126 : RALEIGH ::::
HOST : 129.34.128.245, 129.34.128.246 : YORKTOWN, WATSON ::::
;
```

```
NET  : 9.67.43.0 : RALEIGH.IBM.COM :
;
GATEWAY : 129.34.0.0 : YORKTOWN-GATEWAY ::::
;
```

Using MAKESITE

Because many servers and commands allocate *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO, it is important not to overwrite or delete these data sets while TCP/IP is running. To avoid disrupting any active users, use an *HLQ*=parm that is different than your active *hlq*. This allows you to swap names (by renaming the old HOSTS data sets and then renaming the new HOSTS data sets) without starting and stopping TCP/IP.

Use MAKESITE as a TSO command or in a batch job to generate new *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO data sets. The parameters are the same for either a TSO command or a batch job invocation of MAKESITE. Refer to *z/OS Communications Server: IP System Administrator's Commands* for more information.

After you make changes to your *hlq*.HOSTS.LOCAL data set, you must generate and install new *hlq*.HOSTS.SITEINFO and *hlq*.HOSTS.ADDRINFO data sets.

For the search orders used in locating the local host tables, see "Configuration files for TCP/IP applications" on page 26.

Creating /etc/hosts

The /etc/hosts HFS file can be defined as follows:

- The maximum line length is 256 characters. If a line is greater than 256 characters, it is truncated to 256 characters and processed. If trace resolver is active, a warning message is issued.
- The line starts with an IP address, followed by a blank, followed by host names. Host names are separated by one or more blanks.
- Only IPv4 addresses are supported.
- Each IP address can have up to 35 host names.
- The values for the host name must conform to the following:
  - Maximum of 128 characters.
  - Must contain one or more tokens separated by a period.
  - Each token must be larger than one character and less than 64 characters.
  - First character in each token must start with a letter (A-Z or a-z).
- A comment is indicated by the # or ; character.

For the search orders used in locating /etc/hosts, see "Configuration files for TCP/IP applications" on page 26.

Creating ETC.IPNODES and /etc/ipnodes

The ETC.IPNODES and /etc/ipnodes file can be defined as follows:

- HFS files can reside in any directory. The maximum line length supported is 256 characters. If a line is greater than 256 characters, it is truncated to 256 characters and processed. If trace resolver is active, a warning message is issued.
- MVS data sets must be partitioned organization (PO) or sequential (PS), RECFM=F or RECFM=FB, a logical record length (LRECL) between 56 and 256, and have any valid blocksize (BLKSIZE) for fixed block.

- It can contain IPv4 and IPv6 addresses, but not IPv4 mapped addresses. Each IP address can have up to 35 host names.
- The values for the host name must conform to the following:
  - Maximum of 128 characters.
  - Must contain one or more tokens separated by a period.
  - Each token must be larger than one character and less than 64 characters.
  - First character in each token must start with a letter (A-Z or a-z).
- A comment is indicated by the # or ; character.

The sample IPNODES file provided by z/OS Communications Server follows. It can be found as member EZBREIPN (alias IPNODES) in SEZAINST.

```
;
;  IBM z/OS Communications Server
;  SMP/E distribution name: EZBREIPN
;
;  5694-A01 (C) Copyright IBM Corp. 2002.
;  Licensed Materials - Property of IBM
;
;  Function: Sample ETC.IPNODES file
;
;  The file contains the Internet Protocol (IP) host names
;  and addresses for the local host and other hosts in the
;  Internet network.
;  This file is used to resolve a name into an address (that is, to
;  translate a host name into its Internet address) or resolve
;  an address into a name.
;
;  Comments begin with a # or ; character and continue until the
;  end of the line.
;
;  The following statement defines the Internet Protocol (IP) name
;  and address of the local host and specifies the names and
;  addresses of remote hosts. The maximum line length support is
;  256 characters
;
;  Entries in the hosts file have the following format:
;
;  Address HostName
;
;  Address HostName1 HostName2 HostName3 ..... HostName35
;
;
;  Address: is an IP address, it can be IPV4 or IPV6 address.
;      Note: IPv4-mapped IPv6 address is not allowed.
;  HostName: the length of the hostname is up to 128 characters,
;            and each IP address can have up to 35 hostnames.
;
;
   9.67.43.100 NAMESERVER
   9.67.43.126 RALEIGH
   9.67.43.222 HOSTNAME1.RALEIGH.IBM.COM
   129.34.128.245 YORKTOWN WATSON
   1::2 TESTIPV6ADDRESS1
   1:2:3:4:5:6:7:8 TESTIPV6ADDRESS2
;
```

For the search orders used in locating ETC.IPNODES and /etc/ipnodes, see “Configuration files for TCP/IP applications” on page 26.

Verifying your configuration

At this point, your configuration files have been updated.

To verify a configuration, start the TCP/IP address space and ensure that you see the following message:

```
EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE
```

If the message is not displayed, the messages issued by the TCP/IP address space should describe why TCP/IP did not start.

Verify TCPIP.DATA and TCPIPJOBNAME

Note: For all of the following examples, the unchanged SAMPPROF shipped with TCP/IP is used as the PROFILE.TCPIP. No resolver GLOBALTCPIPDATA was used.

From the TSO ready prompt, verify that the TCPIP.DATA file specifies the correct TCP/IP address space by typing a NETSTAT command. If the wrong TCPIPJOBNAME is specified, you will see the following message:

```
netstat
EZZ2377I Could not establish affinity with TCPXXX (1011/11B3005A) - cannot
         provide the requested option information
READY
```

With a TCPIP.DATA file correctly specifying TCPIP, the following results are displayed. To ensure that the correct TCPIP.DATA file is found in the example, the SYSTCPD is explicitly allocated.

```
alloc f(systcpd) dsn('sys1.tcpparms(tcpdata)') shr reuse
 READY
netstat home
 MVS TCP/IP NETSTAT CS V1R4       TCPIP NAME: TCPIP        17:10:57
 Home address list:
 Address          Link             Flg
 -------          ----             ---
 127.0.0.1        LOOPBACK         P
 READY
```

Verify /etc/resolv.conf

Next, verify the UNIX System Services environment with the onetstat commands. The following example shows the incorrect address space.

The /etc/resolv.conf file is shown.

Note: You only need to create /etc/resolv.conf if you want to maintain separate resolver configuration files for the different APIs.

```
EDIT       /etc/resolv.conf                                 Columns 00001 00072
Command ===>                                                   Scroll ===> PAGE
****** ***************************** Top of Data ******************************
000001 TCPIPJOBNAME TCPCS2

netstat -h
Unable to open UDP socket to TCPCS2   : TCPCS2    is not active.
```

With the /etc/resolv.conf correctly specified with TCPIPJOBNAME TCPIP, the following is displayed:

```
onetstat
MVS TCP/IP onetstat CS V1R4       TCPIP Name: TCPIP         13:15:51
User Id  Conn     Local Socket           Foreign Socket         State
-------  ----     ------------           --------------         -----
BPXOINIT 00000011 0.0.0.0..10007         0.0.0.0..0             Listen
TCPIP    0000000B 0.0.0.0..1025          0.0.0.0..0             Listen
TCPIP    00000010 0.0.0.0..23            0.0.0.0..0             Listen
TCPIP    0000000F 127.0.0.1..1025        127.0.0.1..1026        Establsh
TCPIP    0000000E 127.0.0.1..1026        127.0.0.1..1025        Establsh
Syslogd1 00000012 0.0.0.0..514           *..*                   UDP
```

Verifying PROFILE.TCPIP with netstat or onetstat

Many configuration values specified within the PROFILE.TCPIP file can be verified with the netstat command. To verify the physical network and hardware definitions, use the NETSTAT DEV command. To see operating characteristics use NETSTAT CONFIG. A version of Netstat runs in the TSO and UNIX environments and from the MVS operator environment. Refer to *z/OS Communications Server: IP System Administrator's Commands* for information about the syntax and output of the commands. Following is output from the TSO NETSTAT command. Use the netstat command in the environment with which you are most comfortable.

```
NETSTAT DEVLINKS

MVS TCP/IP NETSTAT CS V1R4       TCPIP NAME: TCPCS            13:40:35
DevName: LOOPBACK          DevType: LOOPBACK
  DevStatus: Ready
  LnkName: LOOPBACK          LnkType: LOOPBACK    LnkStatus: Ready
    NetNum: 0   QueSize: 0
    BytesIn: 2560                  BytesOut: 2560
    ActMtu:  65535
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 0.0.0.0
  Multicast Specific:
    Multicast Capability: No

DevName: LCS1              DevType: LCS       DevNum: 0D00
  DevStatus: Ready
  LnkName: TR1               LnkType: TR          LnkStatus: Ready
    NetNum: 0   QueSize: 0
    BytesIn: 1390158               BytesOut: 842254
    MacAddrOrder: Non-Canonical    SrBridgingCapability: Yes
    IpBroadcastCapability: Yes     ArpBroadcastType: All Rings
    MacAddress:   0123456789AB
    ActMtu:       1492
  BSD Routing Parameters:
    MTU Size: 02000             Metric: 100
    DestAddr: 0.0.0.0           SubnetMask: 255.255.255.128
  Packet Trace Setting:
    Protocol: *                 TrRecCnt: 00000006  PckLength: FULL
    SrcPort: *                  DestPort: *
    IpAddr: *                   SubNet: *
  Multicast Specific:
    Multicast Capability: Yes
    Group             RefCnt
    -----             ------
    224.0.0.1         0000000001

DevName: HYDRAPFD          DevType: MPCIPA
  DevStatus: Ready         CfgRouter: Pri  ActRouter: Pri
  LnkName: LHYDRAF           LnkType: IPAQENET    LnkStatus: Ready
    NetNum: 0   QueSize: 0   Speed: 0000001000
    BytesIn: 0                     BytesOut: 0
    IpBroadcastCapability: No
    ArpOffload: Yes                ArpOffloadInfo: Yes
```

```
    ActMtu:  1000
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 255.0.0.0
  Multicast Specific:
    Multicast Capability: Unknown

DevName: OSATRL90          DevType: ATM
  DevStatus: Not Active
  LnkName: OSA90LINK1        LnkType: ATM         LnkStatus: Not Active
    NetNum: 0   QueSize: 0
    BytesIn: 0                     BytesOut: 0
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 255.0.0.0
  ATM Specific:
    ATM portName:  OSA90
    ATM PVC Name:  STEPH             PVC Status: Not Active

    ATM LIS Name:  LIS1
    SubnetValue:   9.67.1.0          SubnetMask:    255.255.255.0
    DefaultMTU:    0000009180        InactvTimeOut: 0000000300
    MinHoldTime:   0000000060        MaxCalls:      0000001000
    CachEntryAge:  0000000900        ATMArpReTry:   0000000002
    ATMArpTimeOut: 0000000003        PeakCellRate:  0000000000
    NumOfSVCs:     0000000000        BearerClass:   C

    ATMARPSV Name: ARPSV1
    VcType:        PVC               ATMaddrType: NSAP
    ATMaddr:
    IpAddr:        0.0.0.0
  Multicast Specific:
    Multicast Capability: No

DevName: CLAW2             DevType: CLAW      DevNum: 0D10
  DevStatus: Ready         CfgPacking: Yes    ActPacking: Packed
  LnkName: CLAW2LINK         LnkType: CLAW        LnkStatus: Ready
    NetNum: 0   QueSize: 0
    BytesIn: 0                     BytesOut: 0
    ActMtu:  2500
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 255.255.255.0
  Multicast Specific:
    Multicast Capability: Yes
```

The SAMPPROF provided defines only the LOOPBACK address (as shown in this example).

Your installation should have a DEVICE for each interface used by TCP/IP. Counters, BSD Routing Parameters, and Multicast information is given but will not be discussed here. See Chapter 4, “Routing” on page 155 and *z/OS Communications Server: IP Configuration Reference* for more information on these topics.

```
NETSTAT CONFIG

MVS TCP/IP NETSTAT CS V1R4       TCPIP NAME: TCPCS            14:09:59
TCP Configuration Table:
DefaultRcvBufSize:  00016384  DefaultSndBufSize: 00016384
DefltMaxRcvBufSize: 00262144
MaxReTransmitTime:  120.000   MinReTransmitTime: 0.500
RoundTripGain:      0.125     VarianceGain:      0.250
VarianceMultiplier: 2.000     MaxSegLifeTime:    60.000
DefaultKeepALive:   00000120  LogProtoErr:       00
RestrictLowPort:    Yes       SendGarbage:       No
```

```
TcpTimeStamp:       Yes       FinWait2Time:      600

UDP Configuration Table:
DefaultRcvBufSize: 00065535  DefaultSndBufSize: 00065535
CheckSum:          00000001  LogProtoErr:       01
RestrictLowPort:   Yes       NoUdpQueueLimit:   Yes

IP Configuration Table:
   Forwarding: Yes    TimeToLive: 00064  RsmTimeOut:  00060
   FireWall:   No
   ArpTimeout: 01200  MaxRsmSize: 65535  Format:      Short
   IgRedirect: Yes    SysplxRout: Yes    DoubleNop:   No
   StopClawEr: No     SourceVipa: Yes    VarSubnet:   Yes
   MultiPath:  No     PathMtuDsc: Yes    DevRtryDur:  0000000090
   DynamicXCF: Yes
     IpAddr: 199.11.84.104    SubNet: 255.255.248.0    Metric: 00
   IQDIORoute: No
   TcpStackSrcVipa: No

SMF Parameters:
Type 118:
  TcpInit:      01   TcpTerm:    02   FTPClient:    03
  TN3270Client: 00   TcpIpStats: 05
Type 119:
  TcpInit:      No   TcpTerm:    No   FTPClient:    Yes
  TcpIpStats:   Yes  IfStats:    Yes  PortStats:    Yes
  Stack:        Yes  UdpTerm:    Yes  TN3270Client: Yes

Global Configuration Information:
TcpIpStats: 01  ECSALimit: 0002047M  PoolLimit: 2096128K
```

The output from the NETSTAT CONFIG command should show many of the settings specified in PROFILE.TCPIP or implicitly taken from default values. Values set by the PROFILE.TCPIP operating characteristics can be verified at this point.

Verifying interfaces with PING and TRACERTE

PING and TRACERTE can be used to verify adapters or interfaces attached to the z/OS host. Again, oping and otracert can be used in the z/OS UNIX environments with identical results. Since the example shipped with TCPIP has only the LOOPBACK address, for this section a 3172 LCS has been defined.

```
NETSTAT DEVLINKS

MVS TCP/IP NETSTAT CS V1R4       TCPIP NAME: TCPCS            13:40:35
DevName: LOOPBACK          DevType: LOOPBACK
  DevStatus: Ready
  LnkName: LOOPBACK          LnkType: LOOPBACK    LnkStatus: Ready
    NetNum: 0   QueSize: 0
    BytesIn: 2560                  BytesOut: 2560
    ActMtu:  65535
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 0.0.0.0
  Multicast Specific:
    Multicast Capability: No

DevName: LCS1              DevType: LCS       DevNum: 0D00
  DevStatus: Ready
  LnkName: TR1               LnkType: TR          LnkStatus: Ready
    NetNum: 0   QueSize: 0
    BytesIn: 1390158               BytesOut: 842254
    MacAddrOrder: Non-Canonical    SrBridgingCapability: Yes
    IpBroadcastCapability: Yes     ArpBroadcastType: All Rings
    MacAddress:   0123456789AB
    ActMtu:       1492
```

```
  BSD Routing Parameters:
    MTU Size: 02000             Metric: 100
    DestAddr: 0.0.0.0           SubnetMask: 255.255.255.128
  Packet Trace Setting:
    Protocol: *                 TrRecCnt: 00000006  PckLength: FULL
    SrcPort: *                  DestPort: *
    IpAddr: *                   SubNet: *
  Multicast Specific:
    Multicast Capability: Yes
    Group             RefCnt
    -----             ------
    224.0.0.1         0000000001

NETSTAT HOME

MVS TCP/IP NETSTAT CS V1R4       TCPIP NAME: TCPCS            14:15:47
Home address list:
Address          Link             Flg
9.67.113.27      TR1              P
127.0.0.1        LOOPBACK

ping 9.67.113.27
 CS V1R4: Pinging host 9.67.113.27
 Ping #1 response took 0.000 seconds.
 READY
ping 127.0.0.1
 CS V1R4: Pinging host 127.0.0.1
 Ping #1 response took 0.000 seconds.
 READY
tracerte 9.67.113.27
 CS V1R4: Traceroute to 9.67.113.27 (9.67.113.27)
 1 9.67.113.27 (9.67.113.27)  4 ms  6 ms  4 ms
 READY
tracerte 127.0.0.1
 CS V1R4: Traceroute to 127.0.0.1 (127.0.0.1)
 1 LOOPBACK (127.0.0.1)  4 ms  4 ms  4 ms
 READY
```

Given that your PROFILE.TCPIP file contains the interfaces of your installation and that the TCPIP.DATA file contains the correct TCPIPJOBNAME, the TCPIP address space is configured and you can go on to configuring routes, servers, and so on.

Verifying local name resolution with TESTSITE

Use the TESTSITE command to verify that the *hlq*.HOSTS.ADDRINFO and *hlq*.HOSTS.SITEINFO data sets can correctly resolve the name of a host, gateway, or net. For more information on the TESTSITE command, refer to *z/OS Communications Server: IP System Administrator's Commands*.

Verifying PROFILE.TCPIP and TCPIP.DATA using HOMETEST

Use the HOMETEST command to verify the HOSTNAME, DOMAINORIGIN, SEARCH, and NSINTERADDR TCPIP.DATA statements. HOMETEST will use the resolver to obtain the IP addresses assigned to the HOSTNAME and compare them to the HOME list specified in PROFILE.TCPIP. A warning message will be issued if any HOSTNAME IP addresses are missing from the HOME list.

Activate TRACE RESOLVER if you would like detailed information on how the HOSTNAME is resolved to IP addresses. The information will also include what TCPIP.DATA data set names were used. This can be done by issuing the following TSO command before running HOMETEST. The detailed information will be displayed on your TSO screen.

```
allocate dd(systcpt) da(*)
```

Issue the following TSO command after HOMETEST to turn off TRACE RESOLVER output.

```
free dd(systcpt)
```

If you do not have TRACE RESOLVER turned on before running HOMETEST, the following is displayed:

```
hometest

Running IBM MVS TCP/IP CS V1R4 TCP/IP Configuration Tester

FTP.DATA file not found. Using hardcoded default values.

TCP Host Name is: MVS026

Using Name Server to Resolve MVS026
The following IP addresses correspond to TCP Host Name: MVS026
9.67.113.58

The following IP addresses are the HOME IP addresses defined in PROFILE.TCPIP:
9.67.113.58
127.0.0.1

All IP addresses for MVS026 are in the HOME list!

Hometest was successful - all Tests Passed!
```

Verifying your X Windows System installation (Optional)

Note: You cannot verify your X Windows System until after routing and DNS setup. Support is provided for two versions of the X Windows System and the corresponding OSF/Motif. The current support, provided as part of the base IP support in z/OS CS, is for X Windows System Version 11 Release 6 and OSF/Motif Version 1.2. Support for X Windows System Version 11 Release 4 and OSF/Motif Version 1.1 is available as feature HIP614X.

Verifying the X Windows X11R4 System installation

X Windows X11R4 System is installed with the other target libraries. The macro or headers go into the target library data set *hlq*.SEZACMAC. To verify the installation of the X Windows System:

1. Specify your workstation IP address by adding a record (such as the following) to your XWINDOWS.DISPLAY data set.

   ```
   royal.csc.ibm.com:0.0
   ```

   In this example, `royal.csc.ibm.com:0.0` is the name of the host running the X Windows System server.

   Note: No leading blanks are allowed in this record.
2. On the workstation running the X Windows System server, issue an XHOST command specifying the name of your MVS system.
3. Run the program with the XSAMP1 command.

Verifying the X Windows X11R6 System installation

To verify the installation of the X Windows X11R6 System:

1. Ensure that a host (the workstation) with an X Windows System server that supports X11R6 is properly configured and reachable by the MVS system. From the workstation, use Telnet to access the MVS system, and open a z/OS UNIX shell on the MVS system.

2. From the z/OS UNIX shell, export the DISPLAY environment variable using either the network name or the qualified IP address of the workstation as shown in the following example:

   ```
   export DISPLAY=royal.csc.ibm.com:0.0
   ```

   In this example, *royal.csc.ibm.com* is the name of the workstation running the X Windows System server. The display is indicated by *:0.0*, and is specified this way in almost all cases.

3. Authorize the MVS system to access the workstation by executing the XHOST command, and specify either the name of the MVS system or a plus sign (+) as shown in the following example.

   ```
   xhost +
   ```

   Note: The *+* option turns off security for this workstation and allows any X client to display here.

4. The sample X clients are shipped in the directory */usr/lpp/tcpip/X11R6/Xamples/demos*. Change into this directory. There are four sample program directories, xsamp1, xsamp2, xsamp3, and pexsamp. Change to the xsamp1 directory. Verify that there are files named Makefile and xsamp1.c, and then execute the following command:

   ```
   make
   ```

5. Execute the program using the following command:

   ```
   xsamp1
   ```

6. The z/OS UNIX shell should block as another window is opened. Verify the workstation is displaying a new window. The xsamp1 client displays a blank window for 60 seconds and then exits, taking its window with it. The z/OS UNIX shell should no longer be blocked.

Chapter 4. Routing

The objective of this chapter is to guide you through the steps required to configure static or dynamic routing and explain how to verify the configuration. The contents of this chapter are based on the assumption that you understand your entire network configuration. It also assumes that you have read and completed all of the verification tasks outlined in previous chapters in this document.

After reading this chapter, you should be able to do the following:

- Configure static or dynamic routing
- PING a remote host by IP address
- Use TRACERTE to determine the path that will be taken to reach a particular destination
- Use NETSTAT to display your routing table
- Use DISPLAY commands to display dynamic routing information

Note: The definition or modification of an installation's routing configuration should not be performed without a complete understanding of the entire network design.

Routing terminology

The following list describes some of the more common IP routing-related terms and concepts. If you need more detailed information, refer to *Routing in the Internet* by Christian Huitema.

General terms

Autonomous System (AS)
A group of routers exchanging routing information through a common routing protocol. A single AS can represent a large number of IP networks.

Dynamic routes
IP layer routing table entries that are dynamically managed and can automatically change in response to network topology changes. For IPv4, these routes are managed by a routing daemon. For IPv6, these routes are learned by listening to router advertisement messages received from routers.

Exterior Gateway Protocol (EGP)
A routing protocol spoken by routers belonging to different Autonomous Systems when those routers are configured to share routing information between Autonomous Systems. This chapter does not discuss exterior gateway routing.

Interior Gateway Protocol (IGP)
A routing protocol spoken by routers belonging to the same Autonomous System. Each AS has a single IGP. A separate AS within a network can be running a different IGP.

Replaceable static routes
IPv4 static routes that can be replaced by OMPROUTE, or IPv6 static routes that can be replaced by routes learned by listening to router advertisement messages received from routers.

Router
A device or host that interprets protocols at the IP layer and forwards datagrams on a path towards their correct destination.

Routing
The process used in an IP network to deliver a datagram to the correct destination.

Routing daemon
A server process that manages the IP route table.

Static routes
IP layer routing table entries that are manually configured and do not change automatically in response to network topology changes, except when the change is due to an ICMP redirect (if not disabled).

Interior Gateway Protocols (IGP)

An interior gateway protocol is a dynamic route update protocol used between routers that run on TCP/IP hosts within a single autonomous system. The routers use this protocol to exchange information about IP routes.

Some of the more common interior gateway protocols are:

Routing Information Protocol (RIP)
RIP uses a distance vector algorithm to calculate the best path to a destination based on the number of hops in the path. RIP has several limitations. Some of the limitations which exist in RIP Version 1 are resolved by RIP Version 2.

RIP Version 2
RIP Version 2 extends RIP Version 1. Among the improvements are support for multicasting and variable subnetting. Variable subnetting allows the division of networks into variable size subnets. For example, one route can represent addresses from 9.1.1.0 through 9.1.1.255 (the 9.1.1.0/255.255.255.0 subnet) while another can represent addresses from 9.2.0.0 through 9.2.255.255 (the 9.2.0.0/255.255.0.0 subnet).

Open Shortest Path First (OSPF)
OSPF uses a link state or shortest path first algorithm. OSPF's most significant advantage compared to RIP is the reduced time needed to converge after a network change. In general, OSPF is more complicated to configure than RIP and might not be suitable for small networks.

Table 9. Interior Gateway Protocol characteristics

Feature	RIP-1	RIP-2	OSPF
Algorithm	Distance Vector	Distance Vector	Shortest Path First
Network Load (1)	High	High	Low
CPU Processing Requirement (1)	Low	Low	High
IP Network Design Restrictions	Many	Some	Virtually none
Convergence Time	Up to 180 seconds	Up to 180 seconds	Low
Multicast supported (2)	No	Yes	Yes

Table 9. Interior Gateway Protocol characteristics (continued)

Feature	RIP-1	RIP-2	OSPF
Multiple equal-cost routes	No (3)	No(3)	Yes

Notes:

1. Depends on network size and stability.
2. Multicast saves CPU cycles on hosts that are not interested in certain periodic updates, such as OSPF link state advertisements or RIP-2 routing table updates. Multicast frames are filtered out either in the device driver or directly on the interface card if this host has not joined the specific multicast group.
3. RIP in OMPROUTE allows multiple equal-cost routes only for directly-connected destinations over redundant interfaces. See “Using static routing with OMPROUTE” on page 160.

Static versus dynamic routing

Whether static or dynamic routing is used, the IP layer routing mechanism is the same. The IP layer routes a packet by searching its routing table for the most specific route known. Route selection occurs in the following order:

1. If a route exists to the destination address (a host route), it is chosen.
2. At this point, the route chosen depends upon the version of IP being used:
   - For IPv4:
     a. If subnet, network, or supernetwork routes exist to the destination, the route with the most specific network mask (the mask with the most bits on) is chosen.
     b. If the destination is a multicast destination and a multicast default route exists, that route is chosen.
   - For IPv6, if prefix routes exist to the destination, the route with the most specific prefix is chosen.
3. Default routes are chosen when no other route exists to a destination.

Multiple equal-cost routes are allowed for both static and dynamic routing, as depicted in Table 9 on page 156.

The sample network

Figure 33 on page 158 shows a network diagram that depicts a sample network. This sample will be used in the following sections as the configuration of static and dynamic routing is described. See “IPv4 static routing” on page 158, “IPv6 static routing” on page 161, and “Dynamic routing using OMPROUTE” on page 166 for more information.

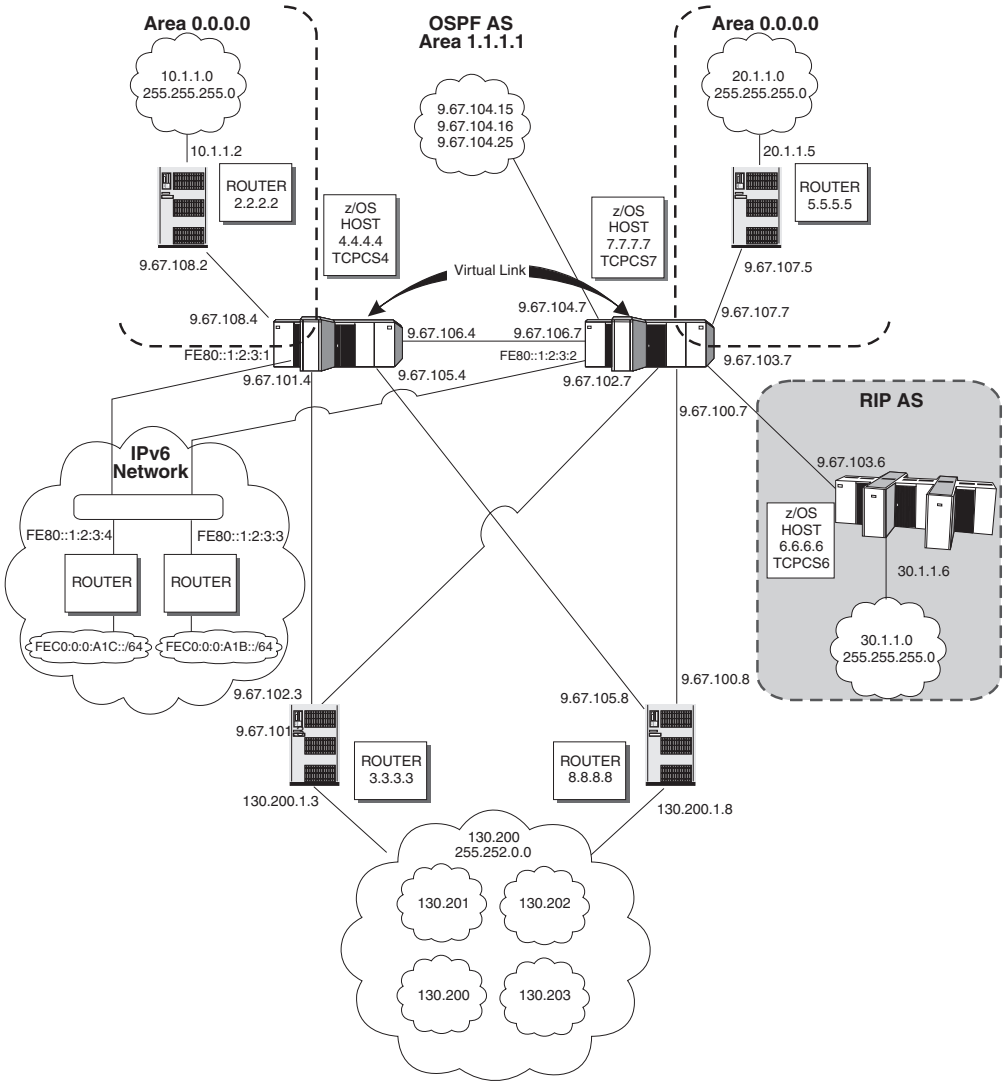


Figure 33. Sample network

Note: In this sample network, TCPCS4 and TCPCS7 are both performing as OSPF Area Border Routers between OSPF Areas 0.0.0.0 and 1.1.1.1. TCPCS7 is also performing as an AS Boundary Router between the OSPF AS and the RIP AS.

IPv4 static routing

Static routing requires that routes are configured manually for each router or destination; this is a significant reason system administrators avoid this technique (if given a choice). Static routing has the disadvantage that network reachability is not dependent on the state of the network itself. If a destination is down or unreachable via that statically configured route, the static routes remain in the routing table, and traffic continues to be sent toward that destination without success.

To minimize network administrator tasks, configuration of static routes is to be avoided, especially in a large network. However, certain circumstances make static routing more appropriate. For example, static routes can be used:

- To define a default route or a route that is not being advertised within a network

- To replace exterior gateway protocols when:
  - Trying to avoid the cost of routing protocol traffic between ASs
  - Trying to avoid complex routing policies
- In conjunction with a routing daemon to provide backup routes when the daemon cannot find a dynamic route to the destination

If static routing is used, only the PROFILE.TCPIP data set has to be updated with either the BEGINROUTES or GATEWAY statements. The BEGINROUTES statement is recommended to define static routes due to its ease of use and additional functionality. Additionally, if static routes are to be replaceable by OMPROUTE, the BEGINROUTES configuration statement must be used. GATEWAY does not support definition of replaceable static routes, and a static route defined on a GATEWAY statement will not be replaceable by a routing daemon.

The only ways to modify static routes are:

- Replace the routing table using the VARY TCPIP,,OBEYFILE command
- Use incoming ICMP Redirect packets
- Use ICMP Must Fragment packets
- If a static route is defined on a BEGINROUTES statement as being replaceable, it can be replaced if a dynamic route is discovered by OMPROUTE. This is the only way that a static route can be replaced by a dynamic route, and a static route cannot be replaced by OROUTED.

For more information on the VARY TCPIP,,OBEYFILE command, the IPCONFIG statement, and the IGNOREREDIRECTS and PATHMTUDISC parameters for the IPCONFIG statement, see *z/OS Communications Server: IP Configuration Reference*.

Note that the first BEGINROUTES or GATEWAY statement in PROFILE.TCPIP or an OBEYFILE data set replaces all static routes in the TCP/IP stack routing table (including those destination addresses specified in the BSDROUTINGPARMS section of the PROFILE.TCPIP). Subsequent statements within the same data set append to the routing table. Also, both BEGINROUTES and GATEWAY statements cannot be used within the same data set.

Every interface must have an IP address to transmit or receive packets. Along with the IP address, each interface must have a subnet mask associated with it for routing purposes. The combination of the address and mask will yield the subnet that the interface belongs to and also determines the broadcast address for the interfaces. There are two ways to specify the subnet mask:

- Specify the netmask on the BSDROUTINGPARMS statement in PROFILE.TCPIP
- Allow z/OS CS to select the interface netmask using information in the routing tables.

The BSDROUTINGPARMS statement is highly recommended to set the netmask value for *each* physical interface. If either OROUTED or NCPROUTE is used, then BSDROUTINGPARMS are required.

Replaceable static routes: Because replaceable static routes are intended to be last-resort routes, TCP/IP attempts to use them only if no dynamic routes to the destination are available.

If a non-replaceable static route fails validation, even if the reason for the failure is transient like gateway unreachable, the definition for the non-replaceable static

route is discarded. However, if a replaceable static route fails validation for a transient reason, the definition of the route is retained and when there are no dynamic routes to the destination, TCP/IP periodically retries, adding the replaceable static route to the routing table. Because of these periodic retries multiple EZZ4333I messages may be seen. Retries will be performed no more often than every 30 seconds, and only as long as there are no active routes to the destination in the routing table, and only if at least one new route has been added to the routing table since the last retry. Retries are terminated as soon as a valid route to the destination is installed into the routing table, whether it is dynamic, static, or replaceable static.

Using static routing with OMPROUTE

It is recommended that non-replaceable static routes not be used with OMPROUTE because this will prevent those routes from being dynamically updated in response to network topology changes. An exception is when routes need to be defined to destinations which, for some reason, will not be learned dynamically via the routing protocol. If static routes are required, use the BEGINROUTES or GATEWAY statement in PROFILE.TCPIP to define them.

TCP/IP will treat static routes defined as replaceable on BEGINROUTES as last-resort routes. These routes can be replaced by dynamic routes. Additionally, if a static route is replaced with a dynamic route, TCP/IP will always retain knowledge of the static route and reinstall it if the destination becomes unreachable using dynamic routes. It is not necessary for TCP/IP to relearn static routes that have been replaced. For this reason, replaceable static routes can be used with OMPROUTE as backup routes, that is, a route to use if nothing is found dynamically.

Another situation where static routes might be required is when multiple, equal-cost routes to a destination are needed and the RIP routing protocol is being used. This is due to the fact that, with the exception of directly attached resources, the RIP protocol will not create multiple, equal-cost routes to a destination. In other words, if multiple adjacent routers are advertising via RIP that they can reach the same destination, RIP will add a route to the TCP/IP route table via only one of those adjacent routers. If it is required that more than one of these routes exist, they would need to be statically configured using the BEGINROUTES or GATEWAY statement in PROFILE.TCPIP. If OROUTED is used instead of OMPROUTE, external entries in the gateways file will be needed. An example of this would be if in Figure 33 on page 158, TCPCS4 used the RIP protocol to Router 3.3.3.3 and Router 8.8.8.8 and if multiple routes were desired to 130.200.0.0 network.

If an installation has multiple interfaces to a directly attached network and it wants to use one interface for input packets and one for output packets (traffic splitting), the installation must use static routes. To do this, a static route could be defined for one and only one interface, forcing all output packets to use that interface. The other routers on the directly attached network would have to be defined with a similar static route, but for the other interface. Although this is the easiest way to implement traffic splitting, if one of the interfaces fails, a host might become unreachable even though the other physical connection may still exist.

Note: A more robust way of accomplishing traffic splitting is to use dynamic routes and make one route preferred over the other via the configured interface costs. See “Step 5: Defining interface costs (OSPF and RIP)” on page 188 for more information.

The BSDROUTINGPARMS statement in PROFILE.TCPIP is not used when the OMPROUTE routing daemon is used. Instead, the interface characteristics, including subnet mask, are defined in the OMPROUTE configuration file.

Note: If you are using NCPROUTE with OMPROUTE, the BSDROUTINGPARMS statement is required to route Transport PDUs prior to OMPROUTE activation. Because the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in the BSDROUTINGPARMS statement and the OMPROUTE configuration file.

IPv6 static routing

Static routing requires that routes are configured manually for each router or destination; this is a significant reason system administrators avoid this technique if given a choice. Static routing has the disadvantage that network reachability is not dependent on the state of the network itself. If a destination is down, or unreachable through a statically configured route, the static routes remain in the routing table and traffic continues to be sent toward that destination without success.

To minimize network administrator tasks, configuration of static routes is to be avoided, especially in a large network. However, certain circumstances make static routing more appropriate. For example, static routes can be used:

- To define a route that will not be learned dynamically from router advertisements received from routers
- In conjunction with dynamic routes to provide backup routes

If static routing is used, only the PROFILE.TCPIP data set has to be updated with BEGINROUTES statements. The only ways to modify static routes are:

- Replace the routing table using the VARY TCPIP,,OBEYFILE command
- Use incoming ICMPv6 redirect packets
- If a static route is defined on a BEGINROUTES statement as being replaceable, it can be replaced by a dynamic route

Notes:

1. The first BEGINROUTES statement in PROFILE.TCPIP or a VARY TCPIP,,OBEYFILE command replaces all static routes in the TCP/IP stack routing table. Subsequent statements within the same data set append to the routing table.
2. If you use static routes and want to honor ICMPv6 redirect messages (that is, you do not code IPCONFIG6 IGNOREREDIRECTS), then you must code the first hop address using the link-local address of the router. This is required since all redirect messages are sent using the router's link-local address, and if the source address of the redirect message does not match the address of the first hop in the routing table, the redirect message will be ignored.

For more information on the VARY TCPIP,,OBEYFILE command, the IPCONFIG6 statement, and the IGNOREREDIRECTS parameter on the IPCONFIG6 statement, see *z/OS Communications Server: IP Configuration Reference*.

Replaceable static routes: Since replaceable static routes are intended to be last-resort routes, TCP/IP only attempts to use them if no dynamic routes to a destination are available. If a non-replaceable static route fails validation, even if the

reason for the failure is transient (for example, gateway unreachable), the definition for the non-replaceable static route is discarded. However, if a replaceable static route fails validation for a transient reason, the definition of the route is retained. When there are no dynamic routes to the destination, TCP/IP periodically retries to add the replaceable static route to the routing table. Because of these periodic retries, multiple EZZ4348I messages might be seen. Retries are performed at the most every 30 seconds, as long as there are no active routes to the destination in the routing table and at least one new route has been added to the routing table since the last retry. Retries are terminated as soon as a valid route to the destination is installed into the routing table, whether it is a dynamic, static, or replaceable static route.

Using static routing with router advertisements

The use of non-replaceable static routes with IPv6 router discovery, when those routes are to destinations that will be learned through received router advertisements, is not recommended. Defining these non-replaceable static routes prevents them from being dynamically updated in response to network topology changes. Examples of routes that are not learned through router advertisements are routes for which the destination address is a specific host address and non-default indirect routes.

TCP/IP treats replaceable static routes as last-resort routes. These routes can be replaced by dynamic (router discovery) routes. In addition, if a static route is replaced with a dynamic route, TCP/IP always retains knowledge of the static route and can reinstall it if the destination becomes unreachable using dynamic routes. It is not necessary for TCP/IP to relearn static routes that have been replaced. For this reason, replaceable static routes can be used with IPv6 router discovery as backup routes, for use if nothing is learned dynamically.

Static routing configuration examples

The following sections illustrate static routing configuration examples.

z/OS TCPCS4

Static route statements for z/OS TCPCS4

```
BEGINRoutes ;first BEGINRoutes in the profile
;
Network/mask                FirstHop      LinkName         PacketSize
Route 9.67.106.0/24         =             CTC4TO7         MTU 1500       ;route1
Route 9.67.105.0/24         =             CTC4TO8         MTU 1500       ;route2
Route 9.67.101.0/24         =             CTC4TO3         MTU 1500       ;route3
Route 9.67.108.0/24         =             CTC4TO2         MTU 1500       ;route4
Route 9.67.107.0/24         9.67.106.7    CTC4TO7         MTU 1500       ;route5
Route 7.7.7.7/32            9.67.106.7    CTC4TO7         MTU 1500       ;route6
Route 9.67.103.0/24         9.67.101.3    CTC4TO3         MTU 1500       ;route7
Route 9.67.103.0/24         9.67.106.7    CTC4TO7         MTU 1500       ;route8
Route 30.1.1.0/24           9.67.106.7    CTC4TO7         MTU 1500       ;route9
Route 10.1.1.0/24           9.67.108.2    CTC4TO2         MTU 1500       ;route10
Route 130.200.0.0/14        9.67.101.3    CTC4TO3         MTU 1500       ;route11
Route 130.200.0.0/14        9.67.105.8    CTC4TO8         MTU 1500       ;route12
Route 130.203.0.0/16        9.67.105.8    CTC4TO8         MTU 1500       ;route13
Route DEFAULT               9.67.106.7    CTC4TO7         MTU 1500       ;route14
;
Destination/PrefixLen       FirstHop      Interface        PacketSize
Route FE80::1:2:3:3/128     =             OSAQDIO46       MTU 5000 REPL  ;route15
Route FE80::1:2:3:4/128     =             OSAQDIO46       MTU 5000 REPL  ;route16
Route FEC0:0:0:A1B::/64     FE80::1:2:3:3 OSAQDIO46       MTU 5000 REPL  ;route17
```

```
Route FEC0:0:0:A1C::/64     FE80::1:2:3:4 OSAQDIO46       MTU 5000 REPL  ;route18
Route DEFAULT6              FE80::1:2:3:4 OSAQDIO46       MTU 5000 REPL  ;route19
EndRoutes
;
```

Notes:

1. In the BEGINROUTES block, the netmask can be specified by a /xx. This number, denoted by xx, represents the number of significant bits in the netmask. For example:

   ```
   /16 = 16 significant bits = 11111111 11111111 00000000 00000000 = 255.255.0.0
   ```

   For IPv6, you must specify the prefix length of the route using the /xxx notation.
2. For direct routes, use an equals symbol (=) for the first hop.

BSDROUTINGPARMS statements for z/OS TCPCS4

```
BSDRoutingParms TRUE ; Shown only for completeness
; Linkname    MTU    Metric    Subnet Mask     Dest Address
  CTC4TO8     1500        0    255.255.255.0   0
  CTC4TO7     1500        0    255.255.255.0   0
  CTC4TO3     1500        0    255.255.255.0   0
  CTC4TO2     1500        0    255.255.255.0   0
  VIPA1A      1500        0    255.255.255.252 0
  EndBSDRoutingParms
;
```

z/OS TCPCS7

Static route statements for z/OS TCPCS7

```
BEGINRoutes
;
Network/mask            FirstHop      LinkName   PacketSize
Route 9.67.106.0/24     =             CTC7TO4   MTU 1500       ;route1
Route 9.67.100.0/24     =             CTC7TO8   MTU 1500       ;route2
Route 9.67.102.0/24     =             CTC7TO3   MTU 1500       ;route3
Route 9.67.103.0/24     =             CTC7TO6   MTU 1500       ;route4
Route 9.67.107.0/24     =             CTC7TO5   MTU 1500       ;route5
Route 4.4.4.4/32        9.67.106.4    CTC7TO4   MTU 1500       ;route6
Route 10.1.1.0/24       9.67.106.4    CTC7TO4   MTU 1500       ;route7
Route 20.1.1.0/24       9.67.107.5    CTC7TO5   MTU 1500       ;route8
Route 30.1.1.0/24       9.67.103.6    CTC7TO6   MTU 1500       ;route9
Route 130.200.0.0/14    9.67.100.8    CTC7TO8   MTU 1500       ;route10
Route 130.200.0.0/14    9.67.102.8    CTC7TO3   MTU 1500       ;route11
Route 130.203.0.0/16    9.67.102.3    CTC7TO3   MTU 1500       ;route12
Route DEFAULT           9.67.107.5    CTC7TO5   MTU 1500       ;route13
;
Destination/PrefixLen   FirstHop      Interface  PacketSize
Route FE80::1:2:3:3/128 =             OSAQDIO76 MTU 5000 REPL  ;route14
Route FE80::1:2:3:4/128 =             OSAQDIO76 MTU 5000 REPL  ;route15
Route FEC0:0:0:A1B::/64 FE80::1:2:3:3 OSAQDIO76 MTU 5000 REPL  ;route16
Route FEC0:0:0:A1C::/64 FE80::1:2:3:4 OSAQDIO76 MTU 5000 REPL  ;route17
Route DEFAULT6          FE80::1:2:3:4 OSAQDIO76 MTU 5000 REPL  ;route18
EndRoutes
```

BSDROUTINGPARMS statements for z/OS TCPCS7

```
BSDRoutingParms TRUE
; Linkname    MTU    Metric    Subnet Mask     Dest Address
  CTC7TO8     1500        0    255.255.255.0   0
  CTC7TO3     1500        0    255.255.255.0   0
  CTC7TO6     1500        0    255.255.255.0   0
  CTC7TO4     1500        0    255.255.255.0   0
```

```
  CTC7TO5     1500        0    255.255.255.0   0
  VIPA1A      1500        0    255.255.255.252 0
  EndBSDRoutingParms
;
```

The sample configuration has an IPv4 supernet route for 130.200.0.0. An IPv4 supernet route means that the netmask for the route is smaller than the class netmask. In this case, 130.200.0.0 is a class B address. The default netmask for class B is 255.255.0.0. The netmask used for this sample is 255.252.0.0, which is less than 255.255.0.0, hence making this a supernet route. In routing, the stack determines a route that has the most bits in common. Therefore, the stack chooses a route in the following order:

1. If a route exists to the destination address (a host route), it is chosen.
2. At this point, the route chosen depends upon the version of IP being used:
   - For IPv4:
     a. If subnet, network, or supernetwork routes exist to the destination, the route with the most specific network mask (the mask with the most bits on) is chosen.
     b. If the destination is a multicast destination and a multicast default route exists, that route is chosen.
   - For IPv6, if prefix routes exist to the destination, the route with the most specific prefix is chosen.
3. Default routes are chosen when no other route exists to a destination.

For example, for TCPCS4 (and when trying to reach 130.200.0.0), route12 in the list is used, which is the supernet route 130.200.0.0 with mask 255.252.0.0. If applying the mask of that route, 255.252.0.0, to the destination IP address, 130.200.0.0, the result is 130.200.0.0, which is the IP address of this route. Now, when trying to reach destination 130.203.5.2, the stack would use route13 in the list, which is a network route for 130.203.00 with mask 255.255.0.0. If applying the mask of that route, 255.255.0.0, to the destination IP address, 130.203.5.2, the result is 130.203.00, which is the IP address of this route.

For TCPCS4, route7 and route8 are examples of equal cost multipath routes to get to 9.67.103.0 subnet. This means that TCPCS4 has two different routes to get to this destination. If IPCONFIG MULTIPATH is not enabled, then only route7 will be used as long as it is active. This is because the stack chooses the first route and ignores route8. If route7 becomes inactive, then the stack will switch and use route8. If MULTIPATH is enabled, then the stack will use both routes according to the MULTIPATH specification.

In the preceding example, all of the IPv4 links have a subnet mask of 255.255.255.0 because this is what is specified for the links in the BSDROUTINGPARMS. Therefore, to determine the broadcast addresses for link CTC4TO3, AND the IP Address, 9.67.101.4, and the subnet mask, 255.255.255.0, to yield the subnet for this link, 9.67.101.0. Then, OR the subnet, 9.67.101.0, with the complement of the subnet mask, 0.0.0.255. This determines that the broadcast address for this link is 9.67.101.255.

For TCPCS4, route15 and route16 would be selected to reach host FE80::1:2:3:3 and host FE80::1:2:3:4 respectively. Route17 and route18 would be selected to reach any IPv6 address that had the first 64 bits of FEC0:0:0:A1B and FEC0:0:0:A1C respectively. Route19 would be selected for any other IPv6 destination.

Notes:

1. All IPv4 IP addresses must follow Classless Inter-Domain Routing (CIDR) convention that requires the actual mask to be one or more on-bits followed by zero or more off-bits. On-bits cannot be followed by off-bits followed by on-bits. Therefore, a mask of 255.255.254.0 is valid (an actual mask of FFFFE00), but a mask of 255.255.253.0 is not valid (an actual mask of FFFFD00) because 253 is 11111101.
2. VIPA links or VIPA interfaces are not allowed on BEGINROUTES statements.
3. You must have a Direct route to a specific IP Address before using that IP Address as the first-hop for indirect routes. A direct route is a route to a destination that is directly connected to the stack by an interface. An indirect route is a route to a destination that is not directly connected, and therefore a router is used to reach that destination. In the preceeding example, for TCPCS4, the subnet route for 9.67.101.0 is directly connected to TCPCS4 by link CTC4T03, and the host route for FE80::1:2:3:3 is directly connected to TCPCS4 by interface OSAQDIO46. However, the subnet route for 9.67.103.0 is indirectly connected and the router used to reach that destination is 9.67.106.7 and/or 9.67.101.3, depending on the MULTIPATH definition.
4. DEFAULT and DEFAULT6 routes are always indirect routes and therefore must always have a first hop address specified.

IPv4 dynamic routing

This section describes the following for IPv4:

- Routing daemons
- Migration from OROUTED to OMPROUTE
- Dynamic routing using OMPROUTE
- Configuring OSPF and RIP

Routing daemons

Daemon is a UNIX term for a background server process. Daemons are used for dynamic routing. For z/OS CS IP, there are two routing daemons:

OROUTED

OROUTED is an IP routing daemon that implements RIP Version 1 and RIP Version 2. It creates and maintains network routing tables. OROUTED determines if a new route has been established or whether a route is temporarily unavailable. For more information, see Appendix E, “Configuring the OROUTED server” on page 769

OMPROUTE

OMPROUTE is an IP routing daemon that supports RIP Version 1, RIP Version 2, and OSPF protocols. You can send RIP Version 1 or RIP Version 2, but not both at the same time on a single interface. However, you can configure a RIP interface to receive both versions. OMPROUTE is the recommended routing daemon application for z/OS CS IP.

Note: OROUTED and OMPROUTE will not run concurrently on the same TCP/IP stack.

Migration from OROUTED to OMPROUTE

An OROUTED start parameter is available to assist with migration from OROUTED to OMPROUTE. This function is invoked by specifying ’-c’ on the OROUTED startup parameters or via the modify command:

```
f orouted,parms='-c'
```

The '-c' parameter uses OROUTED configuration files and OROUTED's current environment (including start parameters and MTU information) to create a file which can be used to create a sample OMPROUTE configuration file. The generated sample contains example configuration statements and lists recommended changes to PROFILE.TCPIP. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details about this start parameter. The file, CNVROUTED.PROFILE (default name) or the name specified by the customer, will be put in the /tmp directory for HFS (an MVS data set is not an option). See the *z/OS Communications Server: IP Configuration Reference* for a comparison of OROUTED configuration statements to OMPROUTE configuration statements.

Refer to the *z/OS Communications Server: IP Migration* for more information on migrating from OROUTED to OMPROUTE.

Dynamic routing using OMPROUTE

OMPROUTE implements the OSPF protocol described in RFC 1583 (OSPF Version 2), the OSPF subagent protocol described in RFC 1850, and the RIP protocols described in RFC 1058 (RIP Version 1) and in RFC 1723 (RIP Version 2). It provides an alternative to the static TCP/IP gateway definitions. The MVS host running with OMPROUTE becomes an active OSPF or RIP router in a TCP/IP network. Either or both of these routing protocols can be used to dynamically maintain the host routing table. For example, OMPROUTE can detect when a route is created, is temporarily unavailable, or if a more efficient route exists. If both OSPF and RIP protocols are used simultaneously, OSPF routes will be preferred over RIP routes to the same destination.

Supported protocols

Open Shortest Path First (OSPF): OSPF is classified as an Interior Gateway Protocol (IGP). This means that it distributes routing information between routers belonging to a single Autonomous System (AS), a group of routers all using a common routing protocol. The OSPF protocol is based on link-state or shortest path first (SPF) technology. It has been designed expressly for the TCP/IP Internet environment, including explicit support for IP subnetting and the tagging of externally-derived routing information.

OSPF performs the following tasks:

Multiple routes
Provides support for multiple equal-cost routes.

Authentication
Provides for the authentication of routing updates.

IP multicast
Uses IP multicast when sending or receiving the updates.

Area routing capability
Area routing capability enables an additional level of routing protection and a reduction in routing protocol traffic.

Allows network grouping
Allows sets of networks to be grouped together. Such a grouping is called an area. The topology of an area is hidden from the rest of the Autonomous System. This method of hiding information enables a significant reduction in routing traffic. Also, routing within the area is determined only by the area's

own topology, lending the area protection from bad routing data. An area is a generalization of an IP subnetted network.

IP subnet configuration
Enables the flexible configuration of IP subnets. Each route distributed by OSPF has a destination and mask. Two different subnets of the same IP network number may have different sizes (that is, different masks). This is commonly referred to as variable length subnetting. A packet is routed to the best (longest or most specific) match. Host routes are considered to be subnets whose masks are all ones (0xFFFFFFFF).

Authenticate OSPF protocol exchanges
Can be configured such that all OSPF protocol exchanges are authenticated. This means that only trusted routers can participate in the Autonomous System's routing. A single authentication scheme is configured for each area. This enables some areas to use authentication while others do not.

OSPF is a dynamic routing protocol. It quickly detects topological changes in the AS (such as router interface failures) and calculates new loop-free routes after a period of convergence. This period of convergence is short and involves a minimum of routing traffic as compared to RIP protocol.

In a link-state routing protocol, each router maintains a database describing the Autonomous System's topology. Each participating router has an identical database. Each individual piece of this database is a particular router's local state (for example, the router's usable interfaces and reachable neighbors). The router distributes its local state throughout the Autonomous System by flooding.

To generate routes, all routers run the exact same algorithm, in parallel. From the topological database, each router constructs a tree of shortest paths with itself as root. This shortest-path tree gives the route to each destination in the Autonomous System. Externally derived routing information appears on the tree as leaves. When several equal-cost routes to a destination exist, the routes (up to four) are added to the TCP/IP stack's route table. The TCP/IP stack uses these equal-cost routes according to the IPCONFIG MULTIPATH statement.

Externally derived routing data (for example, routes learned from the RIP protocol) is passed transparently throughout the Autonomous System. This externally derived data is kept separate from the OSPF protocol's link state data. Each external route can also be tagged by the advertising router, enabling the passing of additional information between routers on the boundaries of the Autonomous System. For information on configuring OSPF, see "Configuring OSPF and RIP" on page 179.

RIP protocol: RIP is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. RIP is based on the Bellman-Ford or the distance-vector algorithm. RIP has many limitations and is not suited for every TCP/IP environment. Before using the RIP function in OMPROUTE, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about RFCs 1058 and 1723.

RIP uses the number of hops, or hop count, to determine the best possible route to a host or network. The term hop count is also referred to as the metric. In RIP, a hop count of 16 means infinity, or that the destination cannot be reached. This limits the longest path in the network that can be managed by RIP to 15 gateways.

A RIP router broadcasts routing information to its directly connected networks every 30 seconds. It receives updates from neighboring RIP routers every 30 seconds and uses the information contained in these updates to maintain the routing table. If an update has not been received from a neighboring RIP router in 180 seconds, a RIP router assumes that the neighboring RIP router is down and sets all routes through that router to a metric of 16 (infinity). If an update has still not been received from the neighboring RIP router after another 120 seconds, the RIP router deletes from the routing table all of the routes through that neighboring RIP router.

RIP Version 2 is an extension of RIP Version 1 and provides the following features:

Route Tags to provide EGP-RIP and BGP-RIP interactions
The route tags are used to separate *internal* RIP routes (routes for networks within the RIP routing domain) from *external* RIP routes, which may have been imported from an EGP (external gateway protocol) or another IGP. OMPROUTE does not generate route tags, but preserves them in received routes and readvertises them when necessary.

Variable subnetting support
Variable length subnet masks are included in routing information so that dynamically added routes to destinations outside subnetworks or networks can be reached.

Immediate next hop for shorter paths
Next hop IP addresses, whenever applicable, are included in the routing information to eliminate packets being routed through extra hops in the network. OMPROUTE will not generate immediate next hops, but will preserve them if they are included in the RIP packets.

Multicasting to reduce load on hosts
IP multicast address 224.0.0.9, reserved for RIP Version 2 packets, is used to reduce unnecessary load on hosts which are not listening for RIP Version 2 messages. This support is dependent on interfaces that are multicast-capable.

Authentication for routing update security
Authentication keys can be configured for inclusion in outgoing RIP Version 2 packets. Incoming RIP Version 2 packets are checked against the configured keys.

Configuration switches for RIP Version 1 and RIP Version 2 packets
Configuration parameters allow for controlling which version of RIP packets are to be sent or received over each interface.

Supernetting support
The supernetting feature is part of Classless InterDomain Routing (CIDR). Supernetting provides a way to combine multiple network routes into fewer supernet routes, thus reducing the number of routes in the routing table and in advertisements.

For configuration information for RIP, see “Configuring OSPF and RIP” on page 179.

OMPROUTE configuration

Run-time environment: OMPROUTE is a z/OS UNIX application, and it requires the Hierarchical File System (HFS) to operate. It can be started from an MVS started procedure, from the z/OS shell, or from AUTOLOG (see “Autolog considerations for OMPROUTE” on page 172 for restrictions on using AUTOLOG to start OMPROUTE). OMPROUTE must be started by an RACF-authorized user ID, and it must reside in an APF authorized library.

OMPROUTE uses the MVS operator's console, SYSLOGD, CTRACE, and STDOUT for its logging and tracing. The MVS operator's console and SYSLOGD are used for major events such as initialization, termination, and error conditions. CTRACE is used for tracing the receipt and transmission of OSPF/RIP packets as well as communications between OMPROUTE and the TCP/IP stack. STDOUT is used for detailed tracing and debugging.

OMPROUTE uses a standard message catalog. The message catalog must be in the HFS. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.

Configuration of OMPROUTE is via an OMPROUTE configuration file. For details on the statements in the OMPROUTE configuration file, refer to *z/OS Communications Server: IP Configuration Reference*.

Display of OMPROUTE information is performed using the DISPLAY command. Modification of OMPROUTE information is performed using the MODIFY command. For details on OMPROUTE's DISPLAY and MODIFY commands, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

Multiple TCP/IP stacks: A one-to-one relationship exists between an instance of OMPROUTE and a stack. OSPF/RIP support on multiple stacks requires multiple instances of OMPROUTE. OMPROUTE and OROUTED cannot run on the same stack concurrently.

TCP/IP stack routing table management: OMPROUTE's job is limited to the management of the TCP/IP stack routing table. OMPROUTE is not involved in the actual routing decisions made by the TCP/IP stack when routing a packet to its destination.

All dynamic routes are deleted from the stack's routing table upon initialization of OMPROUTE. OMPROUTE then repopulates the stack routing table using information learned via the routing protocols.

ICMP Redirects are ignored when OMPROUTE is active.

Unlike OROUTED, OMPROUTE does not make use of the BSDROUTINGPARMS statement. Instead, the Maximum Transmission Unit (MTU), subnet mask, and destination address parameters are configured via the OSPF_INTERFACE, RIP_INTERFACE, and INTERFACE statements in the OMPROUTE configuration file.

Using RIP and OSPF with OMPROUTE: When OMPROUTE is initialized, it uses the OMPROUTE configuration file to determine which routing protocols will be enabled. If at least one OSPF interface is configured, the OSPF protocol is enabled. If at least one RIP interface is configured, RIP is enabled. If OMPROUTE is started with no interfaces defined for a particular protocol, that protocol is disabled until one of the following occurs:

- OMPROUTE is stopped and restarted with a configuration file containing at least one interface of the specific type.
- OMPROUTE is dynamically reconfigured via the MODIFY command with a configuration file containing at least one interface of the specific type.

When OMPROUTE is configured for both the OSPF and RIP protocols, routes that are learned through the OSPF protocol take precedence over routes learned through the RIP protocol.

The OSPF and RIP protocols are communicated over interfaces that are defined with the OSPF_INTERFACE and RIP_INTERFACE configuration statements, respectively. An interface involved in the communication of neither the RIP nor the OSPF protocol should be configured to OMPROUTE via the INTERFACE configuration statement. For non-point-to-point interfaces, an INTERFACE statement is required only to change the default values used by OMPROUTE (for example, to change the default MTU.) OMPROUTE supports a total of 254 interfaces (physical and VIPA). Refer to "VIPA interfaces (Static VIPA and Dynamic VIPA)" on page 185 for special VIPA considerations.

OMPROUTE allows for the generation of multiple, equal-cost routes to a destination. For OSPF and RIP, up to four multiple equal-cost routes are allowed. For RIP, multiple equal-cost routes are supported only to directly connected destinations over redundant interfaces.

Special considerations:

Token-ring multicast: If OMPROUTE will be communicating through the OSPF or RIP Version 2 protocol over a token ring media, and there will be routers attached to that token ring that are not listening (at the DLC layer) for the token ring multicast MAC address 0xC000.0004.0000, the following TRANSLATE statement is required in the PROFILE.TCPIP:

```
TRANSLATE 224.0.0.0 IBMTR FFFFFFFFFFFF linkname
```

Without this statement, OSPF and RIP Version 2 multicast packets are discarded at the DLC layer by those routers that are not listening for the token ring multicast MAC address.

Virtual IP Addresses (VIPA): OMPROUTE is enhanced with Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. The VIPA routes are included in the OSPF and RIP advertisements to adjacent routers. Adjacent routers learn about VIPA routes from the advertisements and can use them to reach the destinations at the MVS host.

Service policy: If service policy is going to be used to restrict access to neighbors on point-to-multipoint interfaces (for example MPCPTP interfaces including XCF and IUTSAMEH connections) for temporary intervals, those neighbors must be explicitly defined on the OSPF_INTERFACE or RIP_INTERFACE statement. Otherwise, OMPROUTE might not be able to communicate with those neighbors when the access restriction expires.

Multiple equal-cost routes: When IPCONFIG MULTIPATH is specified in PROFILE.TCPIP and multiple routes exist in the TCP/IP route table for a destination, outbound traffic for that destination will be spread across all of the routes. This traffic spreading will be done on either a packet-basis or connection-basis depending on the parameter specified on IPCONFIG MULTIPATH. When OMPROUTE is being used to provide dynamic routing for a TCP/IP stack, multiple routes to the same destination can be dynamically added to the TCP/IP stack's route table, based upon the routing information learned from other routers. These multiple routes will be added when the route calculation for each has resulted in the same route cost value. No more than four equal-cost routes will be added for each destination. For RIP, multiple equal-cost routes will be added only to directly-connected destinations over redundant interfaces. The RIP protocol will generate no more than one indirect route to a destination.

Table 10. Multipath route limitations

Multipath route type	BEGINROUTES (Static)	OMPROUTE (OSPF)	OMPROUTE (RIP)
Direct Host	Yes (no limit)	Yes (up to 4)	No
Indirect Host	Yes (no limit)	Yes (up to 4)	No
Direct Network	Yes (no limit)	No	Yes (up to 4 for redundant interfaces)
Indirect Network	Yes (no limit)	Yes (up to 4)	No
Default (Indirect)	Yes (no limit)	Yes (up to 4)	No

Note: Because of the design limitation for multi-access parallel interfaces support, OMPROUTE(OSPF) cannot provide multiple equal-cost network routes that are directly attached to parallel interfaces. However, circumvention would be to define these direct network routes statically in the TCPIP profile using a GATEWAY or BEGINROUTES statement. OMPROUTE will recognize these routes as static equal-cost multipath routes. Also, if more than four equal-cost multipath routes are desired for OSPF or if multiple equal-cost indirect routes are desired for RIP, use the GATEWAY or BEGINROUTES statement.

Configuring OMPROUTE

The steps to configure OMPROUTE are:

1. Create the OMPROUTE configuration file.
2. Reserve the RIP UDP port (if using the RIP protocol).
3. Update the resolver configuration file.
4. Update the OMPROUTE cataloged procedure.
5. Specify the RIP UDP port number in the SERVICES file or data set (if using the RIP protocol).
6. RACF authorize user IDs for starting OMPROUTE.
7. Start syslogd.
8. Update the OMPROUTE environment variables (optional).
9. Create static routes (optional).
10. Configure OSPF Authentication

These steps are described in the following sections.

Step 1: Create the OMPROUTE configuration file: The OMPROUTE configuration file provides information about the host's routing capabilities and TCP/IP interfaces. See "Configuring OSPF and RIP" on page 179 for more detail about the contents of this file. The following is the search order used by OMPROUTE to locate the configuration data set or file:

1. If the environment variable, OMPROUTE_FILE, has been defined, OMPROUTE uses the value as the name of an MVS data set or HFS file to access the configuration data. The syntax for an MVS data set name is `//mvs.dataset.name`. The syntax for an HFS file name is `/dir/subdir/file.name`.
2. `/etc/omproute.conf`
3. hlq.ETC.OMPROUTE.CONF

A sample configuration file is provided in SEZAINST(EZAORCFG). The configuration file for TCPCS4, TCPCS6, and TCPCS7 in the sample network are shown in "Sample OMPROUTE configuration files" on page 202. For a description of the syntax rules for the OMPROUTE configuration file, as well as details on each of the configuration statements, refer to the *z/OS Communications Server: IP Configuration Reference*.

Step 2: Reserve the RIP UDP port (If using the RIP protocol): If the RIP protocol of OMPROUTE is going to be used, UDP port 520 should be reserved for OMPROUTE. This is done by adding the name of the member containing the OMPROUTE cataloged procedure to the PORT statement in PROFILE.TCPIP:

```
PORT
    520 UDP OMPROUTE
```

If you want to be able to start OMPROUTE from the z/OS shell, use the special name OMVS as follows:

```
PORT
   520 UDP OMVS
```

Autolog considerations for OMPROUTE: As discussed in *z/OS Communications Server: IP Configuration Reference*, if a procedure in the AUTOLOG list also has a PORT statement reserving a TCP or UDP port but does not have a listening connection on that port, TCP/IP periodically attempts to cancel that procedure and start it again.

Therefore, if OMPROUTE is being started with AUTOLOG and only the OSPF protocol is being used (no RIP protocol and, therefore, no listening connection on the RIP UDP port), it is important to do one of the following:

- Ensure that the RIP UDP port (520) is not reserved by the PORT statement in the PROFILE.TCPIP.
- Add the NOAUTOLOG parameter to the PORT statement in the PROFILE.TCPIP. For example,

  ```
  PORT
   520 UDP OMPROUTE NOAUTOLOG
  ```

Note: When using only the OSPF protocol, the auto-start feature of AUTOLOG can be used as described above. However, the monitoring and auto-restart features of AUTOLOG are unavailable due to AUTOLOG's dependence on a listening TCP or UDP connection, which does not exist with OSPF.

If you fail to take one of the above actions, OMPROUTE will be periodically canceled and restarted by TCP/IP.

Step 3: Update the resolver configuration file: The resolver configuration file contains keywords (DATASETPREFIX and TCPIPjobname) used by OMPROUTE. The value assigned to DATASETPREFIX will determine the high-level qualifier (*hlq*). The *hlq* is used in the search order for the OMPROUTE configuration file. If no DATASETPREFIX keyword is found, a default of TCPIP is used. The value assigned to TCPIPjobname will be used as the name of the TCP/IP stack with which OMPROUTE establishes a connection.

For a description of the search order used by the resolver to locate the resolver configuration file, see "Resolver configuration files" on page 27.

Step 4: Update the OMPROUTE cataloged procedure: If OMPROUTE is to be started by a procedure, create the cataloged procedure by copying the sample in

SEZAINST(OMPROUTE) to your system or recognized PROCLIB. Specify OMPROUTE parameters and change the data set names to suit your local configuration.

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZBORPRC
//*
//*        5647-A01 (C) Copyright IBM Corp. 1998.
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//OMPROUTE PROC
//OMPROUTE EXEC PGM=OMPROUTE,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")/')
//*
//* Example of start parameters to OMPROUTE:
//*
//* PARM=('POSIX(ON)',
//*       'ENVAR("_CEE_ENVFILE=DD:STDENV")/-t1')
//*
//*        Provide environment variables to run with the
//*        desired stack and configuration.  As an example,
//*        the file specified by STDENV could have these
//*        four lines in it:
//*
//*           RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*           OMPROUTE_FILE=/u/usernnn/config.tcpcs2
//*           OMPROUTE_DEBUG_FILE=/tmp/logs/omproute.debug
//*           OMPROUTE_DEBUG_CONTROL=1000,5
//*
//*        For information on the above environment variables,
//*        refer to the IP CONFIGURATION GUIDE.
//*
//STDENV   DD PATH='/u/usernnn/envcs2',
//         PATHOPTS=(ORDONLY)
//*
//*        The stdout stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stdout file
//*        every time OMPROUTE is started.  If you want to retain
//*        previous stdout information, change it to OAPPEND.
//*
//SYSPRINT DD SYSOUT=*
//*SYSPRINT DD PATH='/tmp/omproute.stdout',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*        The stderr stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stderr file
//*        every time OMPROUTE is started.  If you want to retain
//*        previous stderr information, change it to OAPPEND.
//*
//SYSOUT   DD SYSOUT=*
//*SYSOUT   DD PATH='/tmp/omproute.stderr',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Step 5: Specify the RIP UDP port number in the SERVICES file or data set (If using the RIP protocol): The services file contains the relationship between services and port numbers as described in *z/OS Communications Server: IP Configuration Reference*. The portion of the services file relevant to OMPROUTE is:

```
route           520/udp         router routed
```

The file must exist for the RIP protocol of OMPROUTE to operate.

For a description of the search order used to locate the services file, see
| "Configuration files for TCP/IP applications" on page 26.

Step 6: RACF authorize user IDs for starting OMPROUTE: To reduce risk of an unauthorized user starting OMPROUTE and affecting the contents of the routing table, users who start OMPROUTE must be RACF-authorized to the entity MVS.ROUTEMGR.OMPROUTE and require a UID of zero. To RACF-authorize, the following commands must be entered from a RACF user ID, substituting the authorized user ID on the ID (userid) parameter. The commands in the following example are taken from SEZAINST(EZARACF).

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OMPROUTE) UACC(NONE)
PERMIT MVS.ROUTEMGR.OMPROUTE ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Note: OMPROUTE requires UID=0 for correct installation, configuration, and operation.

Step 7: Start syslogd: To write only the urgent OMPROUTE messages to the z/OS console, syslogd should be running while OMPROUTE is running. Syslogd sends the non-urgent messages to the HFS message log.

Step 8: Update the OMPROUTE environment variables (Optional): The following environment variables are used by OMPROUTE and can be tailored to a particular installation:

RESOLVER_CONFIG
: The RESOLVER_CONFIG variable is used by OMPROUTE to locate the resolver configuration file. For more information on OMPROUTE's use of the resolver configuration file, see "Step 3: Update the resolver configuration file" on page 172. For more information about the RESOLVER_CONFIG environment variable, refer to *z/OS UNIX System Services Planning*.

OMPROUTE_FILE
: The OMPROUTE_FILE variable is used by OMPROUTE in the search order for the OMPROUTE configuration file. For details on the search order used for locating this configuration file, see "Step 1: Create the OMPROUTE configuration file" on page 171.

| **OMPROUTE_OPTIONS**
| : The OMPROUTE_OPTIONS variable is used by OMPROUTE to set various controls for OMPROUTE processing. Currently only the hello_hi option is supported. The syntax of this new variable is:
|
| ```
| OMPROUTE_OPTIONS=hello_hi
| ```
|
| Specifying OMPROUTE_OPTIONS=hello_hi changes the way OMPROUTE processes the OSPF Hello packets. These packets are then given a higher priority than other updates and processed by the first available OMPROUTE task ahead of other received packets. Prior to specifying this parameter,

customers must be cognizant of the impact to their network of processing hello packets out of the received order sequence.

Note: Specifying OMPROUTE_OPTIONS=hello_hi only helps to keep adjacencies up when OMPROUTE is running and getting flooded with protocol packets. It does not provide any help for the case when adjacencies are not staying up because OMPROUTE is not getting enough cycles (that is, swapped out or running in too low a priority).

OMPROUTE_DEBUG_FILE
The OMPROUTE_DEBUG_FILE variable is used by OMPROUTE to override the debug output destination. For more information on using this environment variable, see "OMPROUTE parameters" on page 177.

OMPROUTE_DEBUG_FILE_CONTROL
The OMPROUTE_DEBUG_FILE_CONTROL variable is used by OMPROUTE to control the size and quantity of trace files created when the OMPROUTE_DEBUG_FILE variable is specified. The syntax of this variable is:

```
OMPROUTE_DEBUG_FILE_CONTROL=<size of file>,<num of files>
```

The default values for <size of file> and <num of files> are 200 (kilobytes) and 5 respectively. In general, these values are sufficient for most installations.

Step 9: Create static routes (Optional): OMPROUTE does not use the environment variable GATEWAYS_FILE to initialize static routes. To create static routes, use the BEGINROUTES or GATEWAY statement in PROFILE.TCPIP. For information on the syntax of these statements, see *z/OS Communications Server: IP Configuration Reference*.

During initialization, OMPROUTE learns of static routes by reading the internal routing table set up by TCP/IP. If static routes are changed during execution by VARY TCPIP,,OBEYFILE statements, OMPROUTE is dynamically notified of the changes by TCP/IP. OMPROUTE will advertise active static routes to other routers if allowed by configuration (for example, the IMPORT_STATIC_ROUTES parameter of the AS_BOUNDARY_ROUTING configuration statement).

Static routes can be defined as replaceable or nonreplaceable, with nonreplaceable being the default. A nonreplaceable static route cannot be replaced or modified by OMPROUTE, even if a better dynamic route can be learned and even if the static route is not actually available (but a static route that is not available will not be advertised by OMPROUTE). Because of this, the use of nonreplaceable static routes with OMPROUTE is not recommended unless it is to provide routing over an interface over which no routing protocol is being communicated. A replaceable static route will be replaced by OMPROUTE if it dynamically learns of any other route to the destination. Any dynamically learned route will be considered more desirable than a replaceable static route. A replaceable static route should be considered as a last resort route, to be used by TCP/IP when no dynamic route to a destination can be found. Refer to "Using static routing with OMPROUTE" on page 160 for detailed information.

Step 10: Configure OSPF authentication: OMPROUTE supports defining the OSPF authentication type by area or by interface. All interfaces attached to an area default to the type of authentication defined for that area on the AREA configuration statement, unless overridden on the OSPF_INTERFACE configuration statement. The values of authentication keys must be defined on OSPF_INTERFACE

statements in any case. All routers which could become neighbors of each other must use the same authentication type and key, or OSPF communication between the routers will not be possible.

Virtual links behave similarly to interfaces for authentication purposes. A virtual link will default to use the same type of authentication that is specified for the backbone area unless overridden on the VIRTUAL_LINK configuration statement. When the authentication type is not NONE, the value of the authentication key must be specified on the VIRTUAL_LINK configuration statement. There is no requirement for a virtual link to have the same authentication key value as its underlying real interface.

OSPF authentication does not protect the contents of an OSPF packet. These packets are not encrypted. However, it does provide verification that the packet is genuine.

There are two methods of OSPF authentication: password, and MD5 cryptographic. Password authentication is very basic: an 8-byte password is appended to all OSPF packets and sent in the clear with the rest of the packet. If the sent password matches that defined by the packet receiver, the packet is accepted. MD5 authentication is more sophisticated. The combination of the OSPF packet and the MD5 key is summarized into a 16-byte message digest, which is appended to the packet and sent. The keys are never sent, only the message digests. The receiver then attempts to recreate the message digest from the combination of its defined key and the OSPF packet. If the digest is successfully recreated, the packet is accepted, otherwise it is rejected. MD5 authentication also contains a monotonic increasing counter to protect against replay attacks.

If MD5 cryptographic authentication is being used, a 16-byte MD5 key must be defined on the OSPF_INTERFACE configuration statement. This key is defined as a hexadecimal string and may be obtained in several ways. One method for obtaining MD5 keys is provided in the pwtokey utility, which converts a password into an MD5 key. This Unix System Services utility implements the algorithm defined in RFC 2574. Since OSPF does not support localization of keys, it is only necessary to provide a password to this utility to generate a single, 16-byte key. If multiple sites have this utility, MD5 keys can easily be generated from passwords, which are easier to remember and communicate than 16 byte hexadecimal strings.

Starting and controlling OMPROUTE

After the necessary RACF authorization has been defined (see “Step 6: RACF authorize user IDs for starting OMPROUTE” on page 174), OMPROUTE can be started from an MVS procedure, from the z/OS shell, or from AUTOLOG.

Note: When OMPROUTE is taken down, it should be kept down for at least 3 times the largest dead router interval of the interfaces using MD5 authentication. The same applies to routers adjacent to interfaces using MD5 authentication. Do not stop and start OMPROUTE instantly.

- You can start OMPROUTE from the MVS operators console by starting the OMPROUTE start procedure. A sample start procedure is provided with the product in *hlq*.SEZAINST(OMPROUTE).
- You can start OMPROUTE from the z/OS shell by starting OMVS and then issuing the OMPROUTE command and, optionally, any parameters. For information on parameters, see “OMPROUTE parameters” on page 177.

- You can use the AUTOLOG statement to start OMPROUTE automatically during TCP/IP initialization. Insert the name of the OMPROUTE start procedure in the AUTOLOG statement of the PROFILE.TCPIP data set.

```
AUTOLOG
  OMPROUTE
ENDAUTOLOG
```

  Note: For special considerations when using AUTOLOG to start OMPROUTE, see "Autolog considerations for OMPROUTE" on page 172.

In a Common INET environment, OMPROUTE will attempt to connect to a stack whose name is determined by the TCPIPjobname keyword from the resolver configuration data set or file. In configurations with multiple stacks, a copy of OMPROUTE must be started for each stack that requires OMPROUTE services. To associate OMPROUTE with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique TCPIPjobname.

When running from an MVS procedure, the environment variables can be set by using the STDENV DD statement in the OMPROUTE procedure. For information concerning the environment variables used by OMPROUTE, refer to *z/OS Communications Server: IP Configuration Reference*.

OMPROUTE parameters: OMPROUTE accepts three command line parameters, which govern tracing and debug information. OMPROUTE's trace and debug information is written to stdout with two exceptions:

- When the routing application was started with no tracing, and then a MODIFY command is issued to enable tracing. In this case, the output destination defaults to the file omproute_debug in the current temporary directory (the default is /tmp).
- When the debug output destination has been overridden via the use of an environment variable (OMPROUTE_DEBUG_FILE).

If OMPROUTE is to be started from an MVS procedure, add your parameters to PARM=() in the OMPROUTE cataloged procedure. For example:

```
//* PARM=('POSIX(ON)',
//*       'ENVAR("_CEE_ENVFILE=DD:STDENV")/-t1')
//*
```

If OMPROUTE is to be started from a z/OS shell command line, enter the parameters on the command line.

For either method of starting OMPROUTE, parameters can be specified in mixed case.

Note: Use of the *-tn*, *-dn*, and *-sn* parameters affects OMPROUTE performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

The -tn command line parameter: The *-tn* option specifies the external tracing level, where *n* is a supported trace level. It is intended for customers, testers, service, or developers, and provides information on the operation of the routing application. This option can be used for many purposes, such as debugging a configuration, education on the operation of the routing application, verification of test cases, and so on. The following levels are supported:

1 Informational messages

2 Formatted packet trace

These option levels are cumulative—level 2 includes level 1. For example, -t2 provides formatted packet trace and informational messages.

The -dn and -sn command line parameters: These options specify the internal debugging levels. They are intended for service and provide internal debugging information needed for debugging problems. Use of these parameters can significantly impact performance and are not recommended unless needed to debug a problem. For more information about the use of these parameters, refer to *z/OS Communications Server: IP Diagnosis*.

Controlling OMPROUTE: You can control OMPROUTE from the operator's console using the MODIFY command. The syntax of the MODIFY command can be found in *z/OS Communications Server: IP System Administrator's Commands*. MODIFY commands are available to perform the following functions:

- "Stopping OMPROUTE"
- "Rereading the configuration file"
- "Enabling or disabling the OMPROUTE subagent" on page 179
- "Changing the cost of OSPF links" on page 179
- "Controlling OMPROUTE tracing and debugging" on page 179

Stopping OMPROUTE: OMPROUTE can be stopped in several ways:

- From MVS, issue STOP <procname> or MODIFY <procname>,KILL.

  If OMPROUTE was started from a cataloged procedure, procname is the member name of that procedure. If OMPROUTE was started from the z/OS shell, procname is useridX, where X is the sequence number set by the system. To determine the sequence number, from the SDSF LOG window on TSO, issue /d omvs,u=userid. This will show the programs running under this user ID. The procname can also be set using the environment variable _BPX_JOBNAME and then starting OMPROUTE in the shell background.
- From a z/OS shell superuser ID, issue the kill command to the process ID (PID) associated with OMPROUTE. To determine the PID, use one of the following methods:
  - From the MVS console, issue D OMVS,U=userid, or issue /D OMVS,U=userid at the SDSF LOG window on TSO (where userid is the user ID that started omproute from the shell).
  - Issue the ps -ef command from the z/OS shell.
  - Record the PID when you start OMPROUTE.

For information on the environment variable _BPX_JOBNAME, refer to *z/OS UNIX System Services Planning*. For information on the D OMVS,U=userid command, refer to *z/OS MVS System Commands*.

Rereading the configuration file: The MODIFY <procname>,RECONFIG command is used to reread the OMPROUTE configuration file. This command ignores all statements in the configuration file except new OSPF_INTERFACE, RIP_INTERFACE, and INTERFACE statements. These new configuration statements must be reread from the configuration file through this command prior to the interface being configured to the TCP/IP stack.

Enabling or disabling the OMPROUTE subagent: Use the MODIFY <procname>,ROUTESA=ENABLE command or the MODIFY <procname>,ROUTESA=DISABLE command to enable or disable the OMPROUTE subagent.

Note: To change any other value on the ROUTESA_CONFIG statement, the OMPROUTE application must be recycled.

The OMPROUTE subagent implements RFC 1850 for the OSPF Protocol. The ROUTESA_CONFIG statement is used in the OMPROUTE configuration file to configure the OMPROUTE subagent. For details on ROUTESA_CONFIG, refer to *z/OS Communications Server: IP Configuration Reference*.

Changing the cost of OSPF links: The cost of an OSPF interface can be dynamically changed using the MODIFY <procname>,OSPF,WEIGHT,NAME=<if_name>,COST=<cost> command. This new cost is flooded quickly throughout the OSPF routing domain, and modifies the routing immediately.

The cost of the interface reverts to its configured value whenever the router is restarted. To make the cost change permanent, you must reconfigure the appropriate OSPF_INTERFACE statement in the configuration file.

Controlling OMPROUTE tracing and debugging: The following commands are used to start, stop, or change the level of OMPROUTE tracing and debugging:

- MODIFY <procname>,TRACE=n : for OMPROUTE tracing; n can be 0–2
- MODIFY <procname>,DEBUG=n : for OMPROUTE debugging; n can be 0–4
- MODIFY <procname>,SADEBUG=n : for OMPROUTE subagent debugging; n can be 0 or 1

Note: Use of OMPROUTE tracing and debugging affects OMPROUTE performance and might require increasing the Dead_Router_Interval on OSPF interfaces to keep neighbor adjacencies from collapsing.

Configuring OSPF and RIP

The steps for configuring OSPF and RIP are:

1. Setting the OSPF router ID (If OSPF protocol is used)
2. Defining OSPF areas (If OSPF protocol is used)
3. Limiting information exchange between OSPF areas (If OSPF protocol is used)
4. Defining interfaces (OSPF and RIP)
5. Defining interface costs (OSPF and RIP)
6. Configuring Virtual Links (If OSPF protocol is used)
7. Managing high-cost links (If OSPF protocol is used)
8. Defining filters (If RIP protocol is used)
9. Defining route precedence in a MultiProtocol environment (If OSPF protocol is used)

Step 1: Setting the OSPF router ID (If OSPF protocol is used)

Every router in an OSPF Autonomous System must be assigned a unique router ID. The ROUTERID configuration statement should be coded within the OMPROUTE configuration file to assign the router ID. The value must be one of the OSPF_INTERFACEs defined in the OMPROUTE configuration file. If the ROUTERID configuration statement is not coded, OMPROUTE chooses the IP

address from one of the OSPF_INTERFACE statements as the router ID. With the advent of Dynamic VIPAs (DVIPAs) that can move between z/OS hosts within a sysplex, it is highly recommended that the ROUTERID be a physical interface or a static VIPA, not a Dynamic VIPA.

In the example network shown in Figure 33 on page 158, the ROUTERID is set to the static VIPA address that represents each OMPROUTE router. TCPCS4 has ROUTERID=4.4.4.4, and TCPCS7 has ROUTERID=7.7.7.7.

Step 2: Defining OSPF areas (If OSPF protocol is used)

The sample network shown in Figure 33 on page 158 depicts a network divided using two different methods. The first division is between IP subnetworks within the OSPF Autonomous System (AS) and IP subnetworks external to the OSPF AS (those within the RIP AS). The subnetworks included within the OSPF AS are further subdivided into regions called areas. OSPF areas are collections of contiguous IP subnetworks. The function of areas is to reduce the OSPF overhead required to compute routes to destinations in different areas. Overhead is reduced because less information is exchanged and stored by routers and because fewer CPU cycles are required for a less complex route table calculation.

Every OSPF AS must have at least a backbone area. The backbone is always identified by area number 0.0.0.0. For small OSPF networks, the backbone is the only area required. For larger networks with multiple areas, the backbone provides a core that connects the areas. Unlike other areas, the backbone's subnets can be physically separate. In this case, logical connectivity of the backbone is maintained by configuring virtual links between backbone routers across intervening non-backbone areas. See "Step 6: Configuring Virtual Links (If OSPF protocol is used)" on page 189 for more information on this subject.

Routers that attach to more than one area function as Area Border Routers. All Area Border Routers are part of the backbone, so they must either attach directly to a backbone IP subnet or be connected to another backbone router over a virtual link.

The information and algorithms used by OSPF to calculate routes vary according to whether the destination is within the same area, in a different area within the OSPF AS, or external to the OSPF AS. Every router maintains a database of all links within its area. A shortest path first algorithm is used to calculate the best routes to destinations within the area from this database. Routes between areas are calculated from summary advertisements originated by Area Border Routers for destinations located in other areas of the OSPF AS. External routes (for example, routes to destinations that lie within a RIP AS) are calculated from AS External advertisements originated by AS Boundary Routers and flooded throughout the OSPF AS.

Use the AREA configuration statement to define the areas to which a router attaches. If you do not use the AREA statement, the default is that all OSPF interfaces attach to the backbone area. In the sample network, TCPCS4 and TCPCS7 are both Area Border Routers belonging to both the backbone area (0.0.0.0) and area 1.1.1.1.

```
AREA
  Area_Number=0.0.0.0;

AREA
  Area_Number=1.1.1.1;
```

Step 3: Limiting information exchange between OSPF areas (If OSPF protocol is used)

When Area Border Routers are configured, parameters on the AREA and RANGE configuration statements can be used to control the OSPF route information that crosses the area boundary. For recommendations regarding the usefulness of multiple areas in the z/OS CS environment, refer to “Network design considerations with z/OS CS” on page 193.

One option is to use the AREA statement to define an area as a stub area. AS External advertisements are never flooded into stub areas. In addition, the AREA statement has an option to suppress origination into the stub of summary advertisements for interarea routes. Destinations external to the stub area are still reachable due to the Area Border Routers advertising default routes into stub areas. Traffic within the stub area for unknown destinations is forwarded to the Area Border Router (using the default route). The border router uses its more complete routing information to forward the traffic on an appropriate path toward its destination.

The following requirements must be met for an area to be defined as a stub area:

- No virtual links are configured through the area to maintain backbone connectivity.
- It is acceptable for routers within the area to use a default route for traffic destined outside the AS.
- No routers within the area are AS boundary routers (OSPF routers that advertise routes from external sources as AS External advertisements).

The following AREA statement example meets these requirements:

```
AREA
   Area_Number=2.2.2.2
   Stub_area=Yes
   Import_Summaries=No;
```

Another option is to use IP subnet address ranges to limit the number of summary advertisements originated into an area. A range is defined by an IP address and an address mask. Destinations are considered to fall within the range if the destination address and the range IP address match after the range mask has been applied to both addresses.

When a range is configured for an area at an Area Border Router, the border router suppresses summary advertisements for destinations within that area that fall within the range. The suppressed advertisements would have been originated into the other areas to which the border router attaches. Instead, the Area Border Router may originate a single summary advertisement for the range or no advertisement at all, depending on the option chosen with the RANGE configuration statement.

Notes:

1. If the range is not advertised, there will be no interarea routes for any destination that falls within the range.
2. Ranges cannot be used for areas through which virtual links are configured to maintain backbone connectivity.

In the sample network shown in Figure 33 on page 158, the following RANGE statement could be configured on TCPCS7 to prevent TCPCS7 from advertising destinations in the 9.67.101.0 subnet into the backbone area (Area 0.0.0.0):

```
RANGE
   IP_Address=9.67.101.0
   Subnet_Mask=255.255.255.0
   Area_Number=1.1.1.1
   Advertise=No;
```

Step 4: Defining interfaces (OSPF and RIP)

Each interface in use by the stack should be defined to OMPROUTE using an OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE statement. This section describes the differences between interface types that you should consider when configuring interfaces to OMPROUTE. In general, use the following guidelines:

- An interface over which the OSPF protocol is communicated with other routers must be configured with the OSPF_INTERFACE statement.
- An interface over which the RIP protocol is communicated with other routers must be configured with the RIP_INTERFACE statement.
- All other interfaces should be configured with the INTERFACE statement.

A VIPA interface is an exception to these guidelines and is discussed in more detail in "VIPA interfaces (Static VIPA and Dynamic VIPA)" on page 185.

Communications Server enforces RFC rules against using either a subnetwork's broadcast or network address as a host address. (An address that has all ones in the host portion is a subnet broadcast address. An address that has all zeros in the host portion is the subnet's network address.) Therefore, the subnet_mask on an OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE statement should have enough zero bits such that no home address in that subnet has all zeros or all ones in the host portion of the address. For example if a subnet has two home addresses 10.1.1.1 and 10.1.1.2, then the subnet mask must have zeros in at least two bits; for example, 255.255.255.252. However, if a subnet has four home addresses 10.1.1.1, 10.1.1.2, 10.1.1.3, and 10.1.1.4, then the subnet mask must have zeros in at least three bits; for example, 255.255.255.248; in this case, there could be up to six home addresses in that subnet (10.1.1.1 through 10.1.1.6). In general, if a subnet mask has n zero bits, then there can be up to ((2**n)-2) home addresses in that subnet. This limit applies even if the home addresses are configured on different TCP/IP stacks.

Notes:

1. It is important to define all interfaces to OMPROUTE, even ones not being used for routing because OMPROUTE sets values that override BSDROUTINGPARMS. When an interface that is not defined to OMPROUTE comes up, OMPROUTE will assign it default values, and update the TCP/IP stack's control blocks with these default values, which could result in undesirable effects. These default values include:
   - MTU size set to 576
   - Interface mask set to the class mask. OMPROUTE will generate message EZZ7871 when it does this.
2. OMPROUTE supports up to 254 interfaces (physical and VIPA).

Configuring multi-access parallel interfaces: Whenever configuring multi-access parallel interfaces (primary and secondary redundant interfaces having IP addresses in the same network) for OMPROUTE (OSPF), the order of the parallel interfaces in the HOME list of TCPIP profile must match the order of the corresponding OSPF_INTERFACE statements in the OMPROUTE configuration file. By doing so, OMPROUTE will treat the first interface in the list as primary and the remaining ones as secondaries. The order of the interfaces is critical for OMPROUTE (OSPF) to be able to send the link state updates (LSAs) correctly to

the neighboring routers so that the primary interface can be recognized. Otherwise, a secondary interface configured in OMPROUTE or HOME list may be inadvertently treated as a primary interface and this can cause routing problems between OMPROUTE and its neighbors. In case of failure of a primary interface, OMPROUTE will use the first available secondary interface and mark it as primary.

Note: This procedure is consistent with the method (Method 2) as described in RFC 2178 for OSPF for multi-access parallel interfaces.

Point-to-point (For example CTC and CLAW): For point-to-point interfaces, the destination IP address must be known to OMPROUTE. Specify the DESTINATION_ADDR parameter to allow for the creation of a host route to the address at the remote end of the interface.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.67.106.7
Name=CTC7TO4
Subnet_mask=255.255.255.0
Attaches_to_Area=1.1.1.1
Destination_Addr=9.67.106.4;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.67.103.7
Name= CTC7TO6
Subnet_mask=255.255.255.0
Destination_Addr=9.67.103.6
RIPV2=Yes;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.67.111.1
Name=CTCX
Subnet_mask=255.255.255.0
Destination_addr=9.67.111.2;
```

Note: If another router is directly attached via a CLAW device, and the OSPF protocol is being communicated with that router, the other router must also be configured to view the CLAW device as a point-to-point interface. Failure to do this results in a failure to add any routes via that router.

Point-to-Multipoint: For Point-to-Multipoint capable interfaces (for example MPCPTP interfaces including XCF and IUTSAMEH connections), OMPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate the OSPF or RIP packets. However, due to underlying signaling that takes place when a host connects to these network types, the stack is able to learn the required addresses. In turn, OMPROUTE learns those IP address from the stack. As a result, it is not necessary to configure the IP addresses of the other routers on the interface statements.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.27.13.81
Name=XCFD00
   Attaches_to_Area=1.1.1.1
Subnet_mask=255.255.255.0;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.27.23.81
Name=MPCA01
Subnet_mask=255.255.255.0
RIPV2=Yes;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.27.33.81
Name=XCFB00
Subnet_mask=255.255.255.0;
```

Non-broadcast network interfaces (For example, Hyperchannel and ATM): If the OSPF or RIP protocol communicates with one or more routers over a non-broadcast network interface, OMPROUTE must know the IP addresses of the other routers (neighbors) with which it needs to communicate. For non-broadcast network interfaces, there is no underlying signaling that allows the stack to learn the required IP addresses. As a result, the neighbor addresses must be configured to OMPROUTE with the parameters configured as follows:

- DR_NEIGHBOR and/or the NO_DR_NEIGHBOR parameters on the OSPF_INTERFACE statement
- NEIGHBOR parameter on the RIP_INTERFACE statement
- NON_BROADCAST=YES and ROUTER_PRIORITY parameters on the OSPF_INTERFACE statement

In the OSPF case, DR_NEIGHBOR defines which routers within the non-broadcast network can become the designated router. NO_DR_NEIGHBOR defines which routers cannot become the designated router. ROUTER_PRIORITY defines the priority of this router on the non-broadcast network so that the designated router can be elected for the network. Note that multiple DR_NEIGHBOR and NO_DR_NEIGHBOR parameters can be coded on one statement.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.37.84.49
Name=HCHE00
Subnet_mask=255.255.255.0
   Attaches_to_Area=1.1.1.1
Non_Broadcast=Yes
DR_Neighbor=9.37.84.53
No_DR_Neighbor=9.37.84.63
Cost0=3
Router_Priority=2;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.37.104.79
Name=ATME00
Subnet_mask=255.255.255.0
RIPV2=Yes
Neighbor=9.37.104.85
Neighbor=9.37.104.53;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.77.13.49
Name=ATMB00
Subnet_mask=255.255.255.0;
```

Broadcast network interfaces (For example, Token Ring, Ethernet, and FDDI): When the OSPF or RIP protocol is communicated over a broadcast medium such as Token Ring, Ethernet, or FDDI, these networks allow for broadcasting and multicasting. Therefore, it is not necessary for OMPROUTE to know the IP addresses of the other routers on the network for OSPF or RIP packets to be communicated with those routers. OMPROUTE sends packets to the other routers on the network by using appropriate broadcast or multicast addresses. The IP addresses of the other routers are learned as OSPF/RIP packets are received from them. The OSPF_INTERFACE must include the ROUTER_PRIORITY parameter to assist in electing a Designated Router for the network.

Sample OSPF_INTERFACE

```
OSPF_INTERFACE
IP_Address=9.59.101.5
Name=TR1
   Subnet_mask=255.255.255.0
Attaches_to_Area=1.1.1.1
Cost0=2
Router_Priority=1;
```

Sample RIP_INTERFACE

```
RIP_INTERFACE
IP_Address=9.29.107.3
Name=TR2
Subnet_mask=255.255.255.0
RIPV2=Yes;
```

Sample INTERFACE

```
INTERFACE
IP_Address=9.77.14.49
Name=ETHB00
Subnet_mask=255.255.255.0;
```

If OMPROUTE will be communicating with the OSPF or RIP Version 2 protocol over a token ring media where an attached router does not listen for multicast MAC address 0xC000.0004.0000, see “Token-ring multicast” on page 170.

For interfaces into broadcast media which contain routers that do not support multicast, it is possible to configure the interfaces as Non-Broadcast Network Interfaces. This would cause OMPROUTE to unicast to the neighbor addresses rather than using a multicast address. However, it would also be necessary to configure all the routers on the network to unicast. Otherwise, their multicast packets would never be received.

Note that it is possible to define neighbors using DR_NEIGHBOR and/or NO_DR_NEIGHBOR parameters for OSPF_INTERFACEs and using NEIGHBOR parameters for RIP_INTERFACEs that are broadcast capable, but it is not required or recommended. If you define neighbors on these interfaces, you must define all of them, as OMPROUTE will not communicate RIP or OSPF to undefined neighbors if any are defined on an interface.

VIPA interfaces (Static VIPA and Dynamic VIPA): If only the RIP protocol is used by OMPROUTE, VIPA interfaces should be defined with the INTERFACE statement. If only OSPF or if both OSPF and RIP are used by OMPROUTE, VIPA interfaces should be defined with the OSPF_INTERFACE statement.

Sample OSPF_INTERFACE

```
OSPF example:
OSPF_INTERFACE
IP_Address=4.4.4.4
Name=VIPA1
   Subnet_mask=255.255.255.252;
```

Sample INTERFACE

```
non-OSPF example:
INTERFACE
IP_Address=6.6.6.6
Name=VIPA1
Subnet_mask=255.255.255.252;
```

Note: The most specific subnet mask you can specify is 255.255.255.252.

If the name in an OSPF_INTERFACE or INTERFACE statement refers to a link of type VIRTUAL, then OMPROUTE generates and advertises the following routes whenever applicable:

1. A network route to the network specified in that statement
2. A subnet route to the subnet specified in that statement
3. A host route to the IP_address specified in that statement

Following are the conditions for advertising these routes on a physical network interface to a network:

1. Network route - If VIPA is not in the same network as the physical network interface and is allowed by filters or RANGE.
2. Subnet route - VIPA subnet routes are advertised in OMPROUTE in all conditions, except for RIP when filters prevent it.
3. Host route - as allowed by filters or RANGE. Advertisement of the host route for a VIPA defined on an OSPF_INTERFACE statement can be controlled by the SUBNET parameter on the OSPF_INTERFACE statement that defines that VIPA. If SUBNET=YES, then the host route is not advertised. If SUBNET=NO (the default), the host route is advertised. Care should be taken in using this parameter. VIPA host routes should not be suppressed for dynamic VIPAs or for VIPAs whose subnet might exist on multiple hosts. It is up to the user to ensure these restrictions are enforced, as they are not and cannot be enforced by OMPROUTE.

On the RIP_INTERFACE statement for a physical network interface, the VIPA routes are allowed to be advertised by the following filter parameters:

1. Send_Net_Routes
2. Send_Subnet_Routes
3. Send_Host_Routes, and Send_Only

In addition, the global FILTER and Send_Only statements for RIP can be used to specify which routes are advertised or not.

For OSPF, the RANGE statement can be used to advertise or not to advertise the VIPA routes external to an area in terms of address range based on a subnet mask.

Note: For RIP, the Send_Only = (VIRTUAL) filter in conjunction with the Send_Net_Routes, Send_Subnet_Routes, and Send_Host_Routes filters, or the FILTER statement with VIPA routes, indicates whether or not VIPA routes can be advertised over a RIP interface. Unlike RIP, there are no routing filters for OSPF. For OSPF, the RANGE statement can be used to control which address range of routes can be advertised or not external to an area;

however, it is not granular enough for use as a routing filter. In area-border router configurations, if there are multiple VIPA addresses that are uniquely subnetted, the RANGE statement can be used to specify which VIPA subnet address range of routes can be advertised or not external to an area.

For Dynamic VIPA (DVIPA), link names are assigned programmatically by the stack when the DVIPA is created. Therefore, the name field set on the INTERFACE or OSPF_INTERFACE statement is ignored by OMPROUTE for DVIPAs.

Because a stack could have a large number of DVIPAs defined, as well as DVIPA ranges, additional wildcard capabilities exist on the OSPF_INTERFACE and INTERFACE statements for use only with DVIPAs.

Ranges of DVIPA interfaces can be defined using the Subnet_Mask parameter on the OSPF_INTERFACE or INTERFACE statement. The range defined in this way will be all the IP addresses that fall within the subnet defined by the mask and the IP address. For more information on the Subnet_Mask parameter, see “Step 4: Defining interfaces (OSPF and RIP)” on page 182.

In the example below, DVIPA interfaces in the range of 10.138.65.80 through 10.138.65.95 are defined:

Sample OSPF_INTERFACE

```
OSPF example:
OSPF_INTERFACE
IP_Address=10.138.65.80
Name=DVIPAs
   Subnet_mask=255.255.255.240;
```

Sample INTERFACE

```
non-OSPF example:
INTERFACE
IP_Address=10.138.65.80
Name=DVIPAs
Subnet_mask=255.255.255.240;
```

You must consider an additional issue when VIPAs are being moved between TCP/IP stacks and dynamic routing is provided for those stacks by OMPROUTE. This movement of VIPAs can be done manually or automatically via the use of Dynamic VIPAs. For the VIPAs to be correctly processed and advertised by the routing protocols, they (like all other interfaces) must be configured to OMPROUTE at the time that they become active on the TCP/IP stack. This configuration of VIPAs to OMPROUTE can be accomplished by:

- Explicitly configuring each VIPA with its own OSPF_INTERFACE or INTERFACE statement
- Configuring a range of DVIPAs with a single OSPF_INTERFACE or INTERFACE statement, using the method described above
- Configuring a group of VIPAs with a single OSPF_INTERFACE or INTERFACE statement, using the wildcarding feature available on the interface statements

The recommended approach for configuring OMPROUTE for VIPAs that might move is to preconfigure the OMPROUTE on each TCP/IP stack with all VIPAs that could potentially exist on that stack at some time. Preconfiguring in this way prepares each OMPROUTE for the possible addition of the VIPAs to its stack. During times when the VIPAs do not exist on a particular OMPROUTE’s stack, the configuration information will not be used. However, during periods when the VIPAs

do exist on that OMPROUTE's stack, the configuration information will be available for use by OMPROUTE. This method is recommended because of its ability to respond to movement of the VIPAs between TCP/IP stacks without modification of the OMPROUTE configuration with each move.

If the pre-configuration of VIPAs described in this section has not been done, it is still possible to define a VIPA to OMPROUTE such that it is properly processed and advertised when it becomes active on the corresponding TCP/IP stack. To do this, add the appropriate OSPF_INTERFACE or INTERFACE statement to the OMPROUTE configuration file and then cause OMPROUTE to reread the configuration file by issuing the MODIFY <procname>,RECONFIG command.

Note: You must modify the OMPROUTE configuration file and issue the RECONFIG command prior to the movement of the VIPA to the corresponding TCP/IP stack.

Step 5: Defining interface costs (OSPF and RIP)

Both the OSPF and RIP protocols have a cost value associated with interfaces. With both protocols, the cost of a route to reach a destination is the sum of the costs of each link that will be traversed on the way to the destination. In the sample network shown in Figure 33 on page 158, the cost of a route to get from TCPCS7 to router 3.3.3.3 via TCPCS4 is the cost of the link from TCPCS7 to TCPCS4 plus the cost of the link from TCPCS4 to router 3.3.3.3.

The method for configuring cost values differs between the OSPF and RIP protocols. The cost values of OSPF links, set using the COST0 parameter of the OSPF_INTERFACE statement, should be configured to ensure that preferred routes to destinations will have a lower cost than less preferable routes. The less preferable routes, with the higher cost, will not be used except upon failure of the preferred routes.

For the purpose of the following example, the sample network Figure 33 on page 158 is used and the convention stack (interface) is used to refer to the cost configured for a particular interface on a stack. For instance TCPCS7(9.67.106.7) refers to the cost configured for interface 9.67.106.7 on TCPCS7.

There are three possible routes from TCPCS7 to router 3.3.3.3. They are:

- Direct (TCPCS7 —> 3.3.3.3),
- Via TCPCS4 (TCPCS7 —> TCPCS4 —> 3.3.3.3)
- Via router 8.8.8.8 and TCPCS4 (TCPCS7 —> 8.8.8.8 —> TCPCS4 —> TCPCS3)

If the preferred route from TCPCS7 to router 3.3.3.3 is via TCPCS4, then interface costs must be configured such that the following are true:

```
TCPCS7(9.67.106.7) + TCPCS4(9.67.101.4) < TCPCS7(9.67.102.7)
TCPCS7(9.67.106.7) + TCPCS4(9.67.101.4) < TCPCS7(9.67.100.7) +
8.8.8.8(9.67.105.8) + TCPCS4(9.67.101.4)
```

The reasons for preferring one route over another are numerous. One approach for assigning OSPF link costs would be to set the costs to values inversely proportional to the bandwidth of the physical media. This would result in higher bandwidth routes having lower costs, thus becoming the preferred routes.

The cost values of RIP links are generally set to a value of 1. This results in the cost of a route to a destination being the number of hops to reach the destination. In the sample network, this would result in the three possible RIP routes from TCPCS7 to router 3.3.3.3 having the following costs:

- Direct (TCPCS7 -> 3.3.3.3), cost = 1
- Via TCPCS4 (TCPCS7 -> TCPCS4 -> 3.3.3.3), cost = 2
- Via router 8.8.8.8 and TCPCS4 (TCPCS7 -> 8.8.8.8 -> TCPCS4 -> TCPCS3), cost = 3

If it were desired that the route via TCPCS4 be the preferred route, this could be accomplished by increasing the cost of getting directly from TCPCS7 to router 3.3.3.3. This could be done by increasing either the OUT_METRIC configured on the RIP_INTERFACE statement for 9.67.102.3 on router 3.3.3.3 or the IN_METRIC configured on the RIP_INTERFACE statement for 9.67.102.7 on TCPCS7. Care must be taken when increasing IN_METRIC and OUT_METRIC values to be sure that the cost to reach any destination does not exceed the RIP maximum of 15.

Step 6: Configuring Virtual Links (If OSPF protocol is used)

The OSPF protocol is dependent upon complete connectivity of the backbone area. To maintain backbone connectivity each backbone router must be interconnected. If the configuration of an OSPF Autonomous System is such that the backbone area will become separated into two or more disconnected sections, connectivity must be restored for the protocol to work correctly. This can be done via a Virtual Link. An OSPF Virtual Link should not be confused with a VIPA link. Virtual Links can be configured between any two backbone routers that have an interface to a common non-backbone area. The VIRTUAL_LINK statements specify the ROUTERID of the link endpoint and must be configured at both endpoints. In the sample network shown in Figure 33 on page 158, a Virtual Link is configured between TCPCS4 and TCPCS7 to restore backbone connectivity through Area 1.1.1.1.

Sample TCPCS4

```
TCPCS4:
VIRTUAL_LINK
   Virtual_Endpoint_RouterID=7.7.7.7
Links_Transit_Area=1.1.1.1;
```

Sample TCPCS7

```
TCPCS7:
VIRTUAL_LINK
Virtual_Endpoint_RouterID=4.4.4.4
Links_Transit_Area=1.1.1.1;
```

Step 7: Managing high-cost links (If OSPF protocol is used)

The periodic nature of OSPF routing traffic requires a link's underlying data-link connection to be constantly open. This can result in unwanted usage charges on network segments whose costs are very high. There are two configuration steps that can be taken to inhibit the periodic nature of the protocol.

The first step that can be taken is to define the link as a Demand Circuit. The global Demand_Circuit=YES configuration statement must be specified before any links can be defined as demand circuits. If you configure an OSPF_INTERFACE with the Demand_Circuit=YES parameter, Link State Advertisements (LSAs) sent over the interface will not be periodically refreshed. Only LSAs with real changes will be readvertised. In addition, aging of these LSAs will be disabled such that they will not age out of the link state database.

Another step that can be taken is to define Hello Suppression for the link (using the Hello_Suppression parameter of the OSPF_INTERFACE statement). Hello Suppression is only meaningful if Demand_Circuit=YES and the device is

point-to-point or point-to-multipoint. Refer to *z/OS Communications Server: IP Configuration Reference* for more information on configuring the Hello_Suppression parameter.

If Demand_Circuit=YES and Hello Suppression is implemented, the PP_Poll_Interval parameter of the OSPF_INTERFACE statement can be used to specify the interval at which OMPROUTE should attempt to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available.

Step 8: Defining filters (If RIP protocol is used)

RIP Filters can be configured to OMPROUTE such that certain RIP routing information will not be broadcast out to other routers and/or accepted from other routers. The filters can be applied to individual RIP_INTERFACEs, via the FILTER parameter, or to all RIP interfaces via by the global FILTER statement. When defining a filter, a filter type (sending or receiving) is specified along with a destination/mask address pair. By using filters, an installation can limit the amount of RIP routing information broadcast into the network and/or the amount of RIP routing information maintained by OMPROUTE. In addition, filters can be used to hide destination addresses from portions of the network.

In the sample network shown in Figure 33 on page 158, if you wanted to hide the 10.1.1.0 subnet from TCPCS6 (as well as all routers and hosts on the remote side of TCPCS6), you could define the following filter on TCPCS7:

```
Filter=(nosend,10.1.1.0,255.255.255.0);
```

Step 9: Defining route precedence in a MultiProtocol environment (If OSPF protocol is used)

Note that this discussion of route precedence is quite complicated. If OSPF is the only routing protocol used in your network, route precedence is less of a concern. If, in addition, none of your OSPF routers are configured as AS Boundary Routers, the route precedence concern is entirely eliminated. For environments with multiple protocols or AS Boundary Routers, the following information is provided.

OMPROUTE applies an order of precedence in choosing between two routes to the same destination that were learned via different routing protocols or using information provided by an OSPF AS Boundary Router. To describe this order of precedence applied by OMPROUTE, a few terms must first be defined.

RIP route
: A route learned via the RIP protocol. A RIP route is generated using information provided in a RIP packet from a neighboring router. For example, in the sample network shown in Figure 33 on page 158, the route from TCPCS7 to destination subnet 30.1.1.0 is a RIP route.

OSPF internal route
: A route learned via the OSPF protocol where the entire path traversed to reach the destination lies within the OSPF autonomous system. For example, in the sample network shown in Figure 33 on page 158, the route from TCPCS7 to destination 9.67.108.2 on Router 2.2.2.2 is an OSPF internal route.

OSPF external route
: A route learned via the OSPF protocol where part of the path traversed to reach the destination does not lie within the OSPF autonomous system. The path will leave the autonomous system if it uses information brought into the OSPF autonomous system by an AS Boundary Router. This information brought into the OSPF AS may be information imported from a

different autonomous system (for example, RIP) or information about destinations statically configured on or directly connected to the AS Boundary Router. For example, in the sample network, shown in Figure 33 on page 158, the route from TCPCS4 to destination 9.67.103.6 on TCPCS6 is an OSPF external route. TCPCS7, configured as an AS Boundary Router, has imported information about that destination into the OSPF AS from the RIP AS.

OSPF external routes fall into two categories based upon the setting of the multiprotocol comparison value, which is defined in "MultiProtocol comparison". If the comparison value is set to Type1 on the AS Boundary Router that imports the external information into the OSPF AS, then OSPF external routes generated using this information will be OSPF Type 1 External Routes. If the comparison value is set to Type2 on the AS Boundary Router, then the generated routes will be OSPF Type 2 External Routes. For example, in the sample network, shown in Figure 33 on page 158, if the comparison value on TCPCS7 (an AS Boundary Router) is set to Type 1, the route from TCPCS4 to destination 9.67.103.6 on TCPCS6 is an OSPF Type 1 external route. If the comparison value on TCPCS7 is set to Type 2, the route is an OSPF Type 2 external route.

MultiProtocol comparison: You can configure this comparison value to allow for the specification of how route costs from different autonomous systems should be treated when they coexist. In OMPROUTE, you can configure this value via the COMPARISON configuration statement. When COMPARISON=Type1 is configured, the route cost values used within different autonomous systems (for example, the OSPF AS and the RIP AS) are considered comparable. With COMPARISON=Type2 configured, the route cost values used with the different autonomous systems are considered non-comparable.

The comparison value can be used in several different ways, depending on the function being performed by a router:

- As an AS Boundary Router, OMPROUTE uses the comparison value to determine the type of external routes (Type 1 or Type 2) that is generated by routers in the OSPF AS using routing information that the AS Boundary Router imports into the OSPF AS. See "Step 9: Defining route precedence in a MultiProtocol environment (If OSPF protocol is used)" on page 190 for additional OSPF external route definition information.
- As an AS Boundary Router, OMPROUTE also uses the comparison value in determining how route cost values will be assigned when importing routes from the OSPF AS into the RIP AS.
  - When COMPARISON=Type1 is configured (indicating that cost values are comparable), an OSPF route imported into the RIP AS will be advertised with the actual cost of the OSPF route. For example, in the sample network, if TCPCS7 is configured with COMPARISON=Type1 and the OSPF route from TCPCS7 to destination 9.67.108.2 on TCPCS2 has a cost of 7, then TCPCS7 will advertise into the RIP AS a RIP route to that destination with a cost of 7.

    Notes:

    1. An exception to this rule (defining how OSPF routes are advertised into the RIP AS when COMPARISON=Type1) occurs when the OSPF route to be imported is an OSPF Type 2 External Route. When this is the case, the route is not advertised into the RIP AS at all.
    2. It is important to remember the requirement that all destinations in the RIP AS must be reachable with a cost no greater than 15. Using COMPARISON=Type1 requires that the cost values of OSPF routes be

low. Any destinations in the OSPF AS that can only be reached from the RIP AS with a cost greater than 15 will become unreachable.

  - When COMPARISON=Type2 is configured (indicating that cost values are non-comparable), an OSPF route imported into the RIP AS is advertised with a cost of 1. If a router in the RIP AS has two possible routes to a destination, one internal to the RIP AS and another that was imported from OSPF, this approach results in the route imported from OSPF being favored. For example, in the sample network, Figure 33 on page 158, if TCPCS7 is configured with COMPARISON=Type2 and TCPCS7 can somehow reach a destination in the 30.1.1.0 subnet without passing through TCPCS6 (using links not shown in the sample), then TCPCS7 advertises into the RIP AS a RIP route to the destination with a cost of 1. As a result, TCPCS6 determines that the destination can be reached via TCPCS7 with a cost of 2. If the cost of the route for TCPCS6 to reach the destination internal to the RIP AS is greater than 2, then the route via TCPCS7 is chosen.

    Note: An exception to this rule (defining how OSPF routes are advertised into the RIP AS when COMPARISON=Type2) occurs when the OSPF route to be imported is an OSPF Type 2 External Route. When this is the case, the route is advertised into the RIP AS with the actual cost of the OSPF Type 2 External Route.

- As any router that has routing information from different autonomous systems, OMPROUTE uses the comparison value while choosing between the routes generated using the information from the different autonomous systems. How the comparison value is used in this case is shown in Table 11.

Given these definitions, the order of precedence used in choosing between multiple routes to the same destination, which were learned via the different protocols or by using information provided by an OSPF AS Boundary Router, can be shown in Table 11. In Table 11, *Source Comparison* refers to the setting of the comparison value (using the COMPARISON configuration statement) on the router that is using the order of precedence to choose between the multiple routes, while *Route 1* and *Route 2* are the two possible routes being chosen between.

Table 11. Route precedence

Source comparison	Route 1 Type	Route 2 Type	Route chosen
Type 1	OSPF Internal	RIP	OSPF Internal
Type 1	OSPF Internal	OSPF Type 1 External	OSPF Internal
Type 1	OSPF Internal	OSPF Type 2 External	OSPF Internal
Type 1	RIP	OSPF Type 1 External	Lowest Cost Route
Type 1	RIP	OSPF Type 2 External	RIP Route
Type 1	OSPF Type 1 External	OSPF Type 2 External	OSPF Type 1 External
Type 2	OSPF Internal	RIP	OSPF Internal
Type 2	OSPF Internal	OSPF Type 1 External	OSPF Internal
Type 2	OSPF Internal	OSPF Type 2 External	OSPF Internal

Table 11. Route precedence (continued)

Source comparison	Route 1 Type	Route 2 Type	Route chosen
Type 2	RIP	OSPF Type 1 External	OSPF Type 1 External
Type 2	RIP	OSPF Type 2 External	Lowest Cost Route
Type 2	OSPF Type 1 External	OSPF Type 2 External	OSPF Type 1 External

Network design considerations with z/OS CS

OMPROUTE may be run on z/OS CS for a variety of reasons. If the z/OS CS host is being used as an application or server host and the routing daemon is being run primarily to provide access to network resources, or to provide network resources access to the z/OS CS host, then care must be taken to ensure that the z/OS CS host is not overly burdened with routing work. Unlike routers or other network boxes whose sole purpose is routing, an application host z/OS CS will be doing many things other than routing, and it is not desirable for a large percentage of machine resources (memory and CPU) to be used for routing tasks, as can happen in very complex or unstable networks. In this case the z/OS CS should not be configured as a backbone router, either intentionally or inadvertently. Careful network design can minimize the routing burdens on the z/OS CS application host without compromising the accessibility of z/OS CS resources to the network and vice versa. If care is not taken to minimize the routing work required by the z/OS CS host, OMPROUTE may consume excessive cycles or memory processing huge numbers of routing updates from the network. Or the burden of routing updates may become so large that the z/OS CS cannot keep up because of other workloads on the machine. Since OSPF is heavily timer-driven, this could cause loss of adjacencies and routing problems.

The primary way to reduce the routing burdens on the z/OS CS host is by use of OSPF areas. Refer to “Step 2: Defining OSPF areas (If OSPF protocol is used)” on page 180 for more information. A z/OS CS application host or sysplex can be placed into a non-backbone area with dedicated routers acting as area-border routers. The area-border routers would advertise the z/OS CS’s resources to other attached areas (for example the backbone) and would summarize the network outside the local area to the z/OS CS hosts. If possible, this can be further refined to reduce routing protocol traffic by use of interarea route summarization (accomplished in OMPROUTE area-border routers by the RANGE statement, see *z/OS Communications Server: IP Configuration Reference*, and in Cisco routers with the area range command). Refer to “Step 3: Limiting information exchange between OSPF areas (If OSPF protocol is used)” on page 181 for more information.

An even further, and ideal, optimization would be to make the area containing the z/OS CS application host or sysplex a stub area. In a stub area, only routes within the area are shared among the hosts, and no summaries of other areas are flooded into the area by the area-border routers. Instead, default routes are used to represent all destinations outside the stub area. The stub area’s resources are still advertised to the network at large by the area-border routers. You can only use this optimization if the following two statements apply to your network:

- It is acceptable to use default routes to reach destinations outside the stub area. This means that either there is only one area-border router connecting the stub area to the rest of the network, or if there are multiple such connections they are redundant, so that it does not matter which one is used to get outside the stub area.

- You have no non-OSPF destinations to advertise to the network at large. Stub areas do not permit importation of OSPF external routes. This means for example that you do not have a RIP network attached to the stub area, or if you do, you do not want its destinations reachable from the stub area. Other types of routes that cannot be imported into stub areas include direct routes (for example, for networks attached to interfaces that are not running the OSPF protocol) and static routes. If you define your VIPAs as OSPF_INTERFACE statements in your OMPROUTE configuration file, routes to their addresses will be considered OSPF routes and therefore importable into the stub area and can be advertised by the area-border routers to the network at large.

It is highly recommended to put z/OS CS application hosts or sysplexes into stub areas if at all possible.

A further optimization is to prevent z/OS CS from becoming the designated router on multiaccess media, when pure routers that can perform this function are present. On a multiaccess medium, the designated router and the backup designated router will carry the majority of the routing protocol load for all hosts on the medium. While z/OS CS is capable of performing this role, it does impose additional routing overhead on the system. It would be preferable to allow pure routers to perform this role, if they are available. This is accomplished by ensuring that the pure routers' interfaces onto the medium have higher ROUTER_PRIORITY values than the z/OS CS interfaces on the same medium. However, if the only hosts on a medium are z/OS CS, then one or two of them will have to be designated router or backup designated router.

Verification of OMPROUTE configuration and state

The following sections show sample output from each of the commands that can be used to display OMPROUTE information. The syntax of these DISPLAY commands, as well as detailed information about the data displayed, can be found in *z/OS Communications Server: IP Configuration Reference*.

Note: All commands that include the LIST subparameter indicate that the information being displayed is configured information only and does not necessarily mean that the information is currently being used by OMPROUTE. To display information in current use, use related commands to display current, run-time statistics, and parameters. There are cases when the configured information will not match the in-use information due to some undefined or unresolved information in the OMPROUTE configuration. For example, undefined interfaces or parameters in the OMPROUTE configuration or an incorrect sequence of dynamic reconfiguration with the MODIFY OMPROUTE,RECONFIG command can result in no update of the in-use information at all. Information defined on wildcard interfaces is not displayed in the LIST commands; it is only displayed in the corresponding non-LIST commands when wildcard information is resolved to actual physical interfaces.

Displaying all OSPF configuration information: To display all of the OSPF configuration information, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,ALL
EZZ7831I GLOBAL CONFIGURATION 735
    TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
    STACK AFFINITY:          TCPCS7
    OSPF PROTOCOL:           ENABLED
    EXTERNAL COMPARISON:     TYPE 2
    AS BOUNDARY CAPABILITY:  ENABLED
    IMPORT EXTERNAL ROUTES:  RIP SUB
    ORIG. DEFAULT ROUTE:     NO
```

```
    DEFAULT ROUTE COST:      (1, TYPE 2)
    DEFAULT FORWARD. ADDR.:  0.0.0.0
    DEMAND CIRCUITS:         ENABLED

EZZ7832I AREA CONFIGURATION
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.0          0=NONE           NO          N/A           N/A
1.1.1.1          0=NONE           NO          N/A           N/A

--AREA RANGES--
AREA ID          ADDRESS          MASK             ADVERTISE?
1.1.1.1          9.67.101.0       255.255.255.0    NO

EZZ7833I INTERFACE CONFIGURATION
IP ADDRESS      AREA             COST  RTRNS  TRNSDLY PRI  HELLO   DEAD    DB_EX
7.7.7.7         1.1.1.1             1      5      1     1     10     40      40
9.67.104.7      1.1.1.1             1      5      1     1     10     40      40
9.67.100.7      1.1.1.1             1      5      1     1     10     40      40
9.67.102.7      1.1.1.1             1      5      1     1     10     40      40
9.67.106.7      1.1.1.1             1      5      1     1     10     40      40
9.67.107.7      0.0.0.0             1      5      1     1     10     40      40

EZZ7836I VIRTUAL LINK CONFIGURATION
VIRTUAL ENDPOINT     TRANSIT AREA      RTRNS  TRNSDLY HELLO  DEAD  DB_EX
4.4.4.4              1.1.1.1              10      5     30   180    180

EZZ7835I NBMA CONFIGURATION
                INTERFACE ADDR      POLL INTERVAL
                       9.67.104.7          180
EZZ7834I NEIGHBOR CONFIGURATION
                NEIGHBOR ADDR     INTERFACE ADDRESS   DR ELIGIBLE?
                9.67.104.15       9.67.104.7          YES
                9.67.104.25       9.67.104.7          NO
                9.67.104.16       9.67.104.7
```

Displaying information about configured OSPF areas: To display information about configured OSPF Areas, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,AREAS
EZZ7832I AREA CONFIGURATION 737
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.0          0=NONE           NO          N/A           N/A
1.1.1.1          0=NONE           NO          N/A           N/A

--AREA RANGES--
AREA ID          ADDRESS          MASK             ADVERTISE?
1.1.1.1          9.67.101.0       255.255.255.0    NO
```

Displaying configuration information about configured OSPF interfaces: To display configuration information about configured OSPF interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,IFS
EZZ7833I INTERFACE CONFIGURATION 739
IP ADDRESS      AREA             COST  RTRNS  TRNSDLY PRI  HELLO   DEAD   DB_EX
7.7.7.7         1.1.1.1             1      5      1     1     10     40     40
9.67.104.7      1.1.1.1             1      5      1     1     10     40     40
9.67.100.7      1.1.1.1             1      5      1     1     10     40     40
9.67.102.7      1.1.1.1             1      5      1     1     10     40     40
9.67.106.7      1.1.1.1             1      5      1     1     10     40     40
9.67.107.7      0.0.0.0             1      5      1     1     10     40     40
```

Note: Wildcard interface definitions are not displayed. However, when an actual interface is resolved to a wildcard definition, its information is displayed.

Displaying information about configured Non-broadcast Multiple Access OSPF interfaces: To display information about configured Non-broadcast Multiple Access OSPF interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,NBMA
EZZ7835I NBMA CONFIGURATION 745
               INTERFACE ADDR      POLL INTERVAL
               9.67.104.7          180
```

Displaying information about configured OSPF Virtual Links: To display information about configured OSPF virtual links, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,VLINKS
EZZ7836I VIRTUAL LINK CONFIGURATION 747
VIRTUAL ENDPOINT     TRANSIT AREA      RTRNS  TRNSDLY HELLO  DEAD  DB_EX
4.4.4.4              1.1.1.1              10      5     30   180    180
```

Displaying information about configured OSPF neighbors: To display information about configured OSPF neighbors enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LIST,NBRS
EZZ7834I NEIGHBOR CONFIGURATION 749
               NEIGHBOR ADDR     INTERFACE ADDRESS   DR ELIGIBLE?
               9.67.104.15       9.67.104.7          YES
               9.67.104.25       9.67.104.7          NO
               9.67.104.16       9.67.104.7          NO
```

Displaying the contents of a single OSPF link state advertisement: To display the contents of a single OSPF link state advertisement, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,LSA,LSTYPE=1,LSID=7.7.7.7,ORIG=7.7.7.7,AREAID=1.1.1.1
EZZ7880I LSA DETAILS 751
        LS AGE:          521
        LS OPTIONS:      E,DC
        LS TYPE:         1
        LS DESTINATION (ID): 7.7.7.7
        LS ORIGINATOR:   7.7.7.7
        LS SEQUENCE NO:  0X80000013
        LS CHECKSUM:     0XA9A
        LS LENGTH:       120
        ROUTER TYPE:  ABR,ASBR,V
        # ROUTER IFCS:   8
                LINK ID:          7.7.7.4
                LINK DATA:        255.255.255.252
                INTERFACE TYPE:   3
                        NO. OF METRICS: 0
                        TOS 0  METRIC:  1
                LINK ID:          8.8.8.8
                LINK DATA:        9.67.100.7
                INTERFACE TYPE:   1
                        NO. OF METRICS: 0
                        TOS 0 METRIC:   1 (1)
                LINK ID:          3.3.3.3
                LINK DATA:        9.67.102.7
                INTERFACE TYPE:   1
                        NO. OF METRICS: 0
                        TOS 0 METRIC:   1 (1)
                LINK ID:          4.4.4.4
                LINK DATA:        9.67.106.7
                INTERFACE TYPE:   1
                        NO. OF METRICS: 0
                        TOS 0 METRIC:   1 (1)
                LINK ID:          7.7.7.7
                LINK DATA:        255.255.255.255
                INTERFACE TYPE:   3
                        NO. OF METRICS: 0
```

```
                        TOS 0 METRIC:   1
                LINK ID:          9.67.100.8
                LINK DATA:        255.255.255.255
                INTERFACE TYPE:   3
                        NO. OF METRICS: 0
                        TOS 0 METRIC:   1
                LINK ID:          9.67.102.3
                LINK DATA:        255.255.255.255
                INTERFACE TYPE:   3
                        NO. OF METRICS: 0
                        TOS 0 METRIC:   1
                LINK ID:          9.67.106.4
                LINK DATA:        255.255.255.255
                INTERFACE TYPE:   3
                        NO. OF METRICS: 0
                        TOS 0 METRIC:   1
```

Displaying statistics and parameters for OSPF areas: To display statistics and parameters for all OSPF areas attached to the router, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,AREASUM
EZZ7848I AREA SUMMARY 757
AREA ID        AUTHENTICATION   #IFCS  #NETS  #RTRS  #BRDRS DEMAND
0.0.0.0           NONE             2      0      4      2  ON
1.1.1.1           NONE             5      0      4      2  ON
```

Displaying the list of AS external advertisements: To display a list of AS external advertisements that are in the OSPF link state database, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,EXTERNAL
EZZ7853I AREA LINK STATE DATABASE 759
TYPE LS DESTINATION     LS ORIGINATOR     SEQNO     AGE   XSUM
  5 @6.6.6.6            7.7.7.7         0X80000007  825  0X1B5C
  5 @9.67.103.6         7.7.7.7         0X80000007  831  0XE1F3
  5 @10.1.1.0           2.2.2.2         0X80000003 1690  0X2775
  5 @10.1.1.1           2.2.2.2         0X80000003 1690  0X1D7E
  5 @20.1.1.0           5.5.5.5         0X80000003 1616  0X4A3C
  5 @20.1.1.1           5.5.5.5         0X80000003 1616  0X4045
  5 @30.0.0.0           7.7.7.7         0X80000006  831  0XB0C0
  5 @30.1.1.0           7.7.7.7         0X80000006  831  0X99D5
  5 @30.1.1.4           7.7.7.7         0X80000001  825  0X7BF4
  5 @30.1.1.8           7.7.7.7         0X80000001  825  0X5319
  5 @130.200.0.0        3.3.3.3         0X80000003 1695  0X98C0
  5 @130.200.0.0        8.8.8.8         0X80000003 1630  0X243
  5 @130.200.1.1        3.3.3.3         0X80000003 1695  0X83D3
  5 @130.200.1.18       8.8.8.8         0X80000003 1630  0X42EF
  5 @130.201.0.0        3.3.3.3         0X80000003 1695  0X8CCB
  5 @130.201.0.0        8.8.8.8         0X80000003 1630  0XF54E
  5 @130.202.0.0        3.3.3.3         0X80000003 1694  0X80D6
  5 @130.202.0.0        8.8.8.8         0X80000003 1629  0XE959
               # ADVERTISEMENTS:       18
               CHECKSUM TOTAL:         0X83472
```

Displaying a list of non-AS external advertisements: To display a list of non-AS external advertisements that are in the OSPF link state database for a particular OSPF area, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,DATABASE,AREAID=1.1.1.1
EZZ7853I AREA LINK STATE DATABASE 761
TYPE LS DESTINATION     LS ORIGINATOR     SEQNO     AGE   XSUM
  1 @3.3.3.3            3.3.3.3         0X8000000F  879  0X8B11
  1 @4.4.4.4            4.4.4.4         0X8000001A  713  0XA020
  1 @7.7.7.7            7.7.7.7         0X80000013  711  0XA9A
  1 @8.8.8.8            8.8.8.8         0X8000000D  861  0XBD81
  3 @2.2.2.2            4.4.4.4         0X80000003 1676  0XC45C
```

```
  3 @5.5.5.4           7.7.7.7         0X80000003  880  0XE327
  3 @5.5.5.5           7.7.7.7         0X80000003  880  0XDF29
  3 @7.7.7.4           7.7.7.7         0X80000001  710  0X956E
  3 @9.67.107.5        7.7.7.7         0X80000006  881  0X4A14
  3 @9.67.107.7        7.7.7.7         0X80000003  880  0X4618
  3 @9.67.108.2        4.4.4.4         0X80000003 1667  0XBDB1
  3 @9.67.108.4        4.4.4.4         0X80000003 1658  0XB3B8
  4 @2.2.2.2           4.4.4.4         0X80000003 1658  0XAC74
  4 @5.5.5.5           7.7.7.7         0X80000003  880  0XC741
               # ADVERTISEMENTS:       14
               CHECKSUM TOTAL:         0X884B0
```

Displaying current, run-time statistics and parameters for OSPF interfaces: To display current, run-time statistics and parameters for OSPF interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,INTERFACE
EZZ7849I INTERFACES 763
IFC ADDRESS     PHYS         ASSOC. AREA     TYPE   STATE  #NBRS  #ADJS
7.7.7.7         VIPA1A       1.1.1.1         VIPA    N/A    N/A    N/A
9.67.104.7      NBMA7        1.1.1.1         MULTI     1      3      0
9.67.100.7      CTC7TO8      1.1.1.1         P-P      16      1      1
9.67.102.7      CTC7TO3      1.1.1.1         P-P      16      1      1
9.67.106.7      CTC7TO4      1.1.1.1         P-P      16      1      1
9.67.107.7      CTC7TO5      0.0.0.0         P-P      16      1      1
UNNUMBERED      VL/0         0.0.0.0         VLINK    16      1      1
```

Displaying current, run-time statistics and parameters for a specific OSPF interface: To display current, run-time statistics and parameters for a specific OSPF interface, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,IF,NAME=CTC7TO4
EZZ7850I INTERFACE DETAILS 769
               INTERFACE ADDRESS:      9.67.106.7
               ATTACHED AREA:          1.1.1.1
               PHYSICAL INTERFACE:     CTC7TO4
               INTERFACE MASK:         255.255.255.0
               INTERFACE TYPE:         P-P
               STATE:                  16
               DESIGNATED ROUTER:      0.0.0.0
               BACKUP DR:              0.0.0.0

  DR PRIORITY:       1  HELLO INTERVAL:   10  RXMT INTERVAL:     5
  DEAD INTERVAL:    40  TX DELAY:          1  POLL INTERVAL:     0
  DEMAND CIRCUIT:  OFF  HELLO SUPPRESS:  OFF  SUPPRESS REQ:    OFF
  MAX PKT SIZE:   1024  TOS 0 COST:        1  AUTHTYPE:   PASSWORD

  # NEIGHBORS:       1  # ADJACENCIES:     1  # FULL ADJS.:      1
  # MCAST FLOODS:   15  # MCAST ACKS:      4  DL UNICAST:      OFF
  MC FORWARDING:   OFF

  NETWORK CAPABILITIES:
   POINT-TO-POINT
   DEMAND-CIRCUITS
```

Displaying current, run-time statistics and parameters for OSPF neighbors: To display current, run-time statistics and parameters for OSPF neighbors, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,NBR
EZZ7851I NEIGHBOR SUMMARY 771
NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
9.67.104.16     0.0.0.0             1     0     0     0  OFF NBMA7
9.67.104.25     0.0.0.0             1     0     0     0  OFF NBMA7
9.67.104.15     0.0.0.0             1     0     0     0  OFF NBMA7
9.67.100.8      8.8.8.8           128     0     0     0  OFF CTC7TO8
9.67.102.3      3.3.3.3           128     0     0     0  OFF CTC7TO3
```

```
9.67.106.4      4.4.4.4           128     0     0     0  OFF CTC7TO4
9.67.107.5      5.5.5.5           128     0     0     0  OFF CTC7TO5
VL/0            4.4.4.4           128     0     0     0  OFF *
```

Displaying current run-time statistics and parameters for a specific OSPF neighbor: To display current run-time statistics and parameters for a specific OSPF neighbor, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,NBR,IPADDR=9.67.106.4
EZZ7852I NEIGHBOR DETAILS 779
                NEIGHBOR IP ADDRESS:    9.67.106.4
                OSPF ROUTER ID:         4.4.4.4
                NEIGHBOR STATE:         128
                PHYSICAL INTERFACE:     CTC7TO4
                DR CHOICE:              0.0.0.0
                BACKUP CHOICE:          0.0.0.0
                DR PRIORITY:            1
                NBR OPTIONS:            E
  DB SUMM QLEN:      0  LS RXMT QLEN:      0  LS REQ QLEN:       0
  LAST HELLO:        4  NO HELLO:        OFF
  # LS RXMITS:       1  # DIRECT ACKS:     0  # DUP LS RCVD:     6
  # OLD LS RCVD:     0  # DUP ACKS RCVD:   1  # NBR LOSSES:      0
```

Displaying routes to other routers that have been calculated by OSPF: To display routes to other routers that have been calculated by OSPF, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,ROUTERS
EZZ7855I OSPF ROUTERS 781
DTYPE RTYPE DESTINATION       AREA           COST       NEXT HOP(S)
ASBR  SPF   2.2.2.2          0.0.0.0         2         9.67.106.4
  BR  SPF   4.4.4.4          0.0.0.0         1         9.67.106.4
ASBR  SPF   5.5.5.5          0.0.0.0         1         9.67.107.5
ASBR  SPF   3.3.3.3          1.1.1.1         1         9.67.102.3
  BR  SPF   4.4.4.4          1.1.1.1         1         9.67.106.4
ASBR  SPF   8.8.8.8          1.1.1.1         1         9.67.100.8
```

Displaying the number of LSAs currently in the link state database: To display the number of LSAs currently in the link state database, categorized by type, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,DBSIZE
EZZ7854I LINK STATE DATABASE SIZE 783
                # ROUTER-LSAS:            8
                # NETWORK-LSAS:           0
                # SUMMARY-LSAS:           37
                # SUMMARY ROUTER-LSAS:    7
                # AS EXTERNAL-LSAS:       18
                # INTRA-AREA ROUTES:      24
                # INTER-AREA ROUTES:      1
                # TYPE 1 EXTERNAL ROUTES: 0
```

Displaying statistics generated by the OSPF routing protocol: To display statistics generated by the OSPF routing protocol, enter the following command:

```
D TCPIP,TCPCS7,OMP,OSPF,STATS
EZZ7856I OSPF STATISTICS 785
                OSPF ROUTER ID:         7.7.7.7
                EXTERNAL COMPARISON:    TYPE 2
                AS BOUNDARY CAPABILITY: YES
                IMPORT EXTERNAL ROUTES: RIP SUB
                ORIG. DEFAULT ROUTE:    NO
                DEFAULT ROUTE COST:     (1, TYPE 2)
                DEFAULT FORWARD. ADDR.: 0.0.0.0
ATTACHED AREAS:                 2  OSPF PACKETS RCVD:              821
OSPF PACKETS RCVD W/ERRS:       0  TRANSIT NODES ALLOCATED:         55
TRANSIT NODES FREED:           47  LS ADV. ALLOCATED:               263
```

```
LS ADV. FREED:                 201  QUEUE HEADERS ALLOC:             96
QUEUE HEADERS AVAIL:            96  MAXIMUM LSA SIZE:               976
# DIJKSTRA RUNS:                9  INCREMENTAL SUMM. UPDATES:        4
INCREMENTAL VL UPDATES:          0  MULTICAST PKTS SENT:            746
UNICAST PKTS SENT:             107  LS ADV. AGED OUT:                 0
LS ADV. FLUSHED:                22  PTRS TO INVALID LS ADV:           0
INCREMENTAL EXT. UPDATES:       49
```

Displaying the routes in the OMPROUTE routing table: To display all of the routes in the OMPROUTE routing table, enter the following command:

```
D TCPIP,TCPCS7,OMP,RTTABLE
EZZ7847I ROUTING TABLE 796
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SBNT   2.0.0.0          FF000000  1       1368    NONE
 SPF   2.2.2.0          FFFFFFFC  3       1380    9.67.106.4
 SPF   2.2.2.2          FFFFFFFF  3       1380    9.67.106.4
SBNT   3.0.0.0          FF000000  1       1549    NONE
 SPF   3.3.3.0          FFFFFFFC  2       1561    9.67.102.3
 SPF   3.3.3.3          FFFFFFFF  2       1561    9.67.102.3
SBNT   4.0.0.0          FF000000  1       1549    NONE
 SPF   4.4.4.4          FFFFFFFC  2       1561    9.67.106.4
 SPF   4.4.4.4          FFFFFFFF  2       1561    9.67.106.4
SBNT   5.0.0.0          FF000000  1       1549    NONE
 SPF   5.5.5.4          FFFFFFFC  2       1567    9.67.107.5
 SPF   5.5.5.5          FFFFFFFF  2       1567    9.67.107.5
SBNT   6.0.0.0          FF000000  1       1549    NONE
 RIP   6.6.6.4          FFFFFFFC  2       30      9.67.103.6
SBNT   7.0.0.0          FF000000  1       1368    NONE
SPIA*  7.7.7.4          FFFFFFFC  3       1380    9.67.106.4
 DIR*  7.7.7.7          FFFFFFFF  1       1574    VIPA1A
SBNT   8.0.0.0          FF000000  1       1549    NONE
 SPF   8.8.8.8          FFFFFFFC  2       1545    9.67.100.8
 SPF   8.8.8.8          FFFFFFFF  2       1545    9.67.100.8
SBNT   9.0.0.0          FF000000  1       1368    NONE
 DIR*  9.67.100.0       FFFFFF00  1       1576    9.67.100.7
 SPF   9.67.100.7       FFFFFFFF  2       1545    CTC7TO8
 SPF   9.67.100.8       FFFFFFFF  1       1572    9.67.100.8
 SPF   9.67.101.3       FFFFFFFF  2       1561    9.67.106.4
 SPF   9.67.101.4       FFFFFFFF  2       1561    9.67.102.3
 DIR*  9.67.102.0       FFFFFF00  1       1575    9.67.102.7
 SPF   9.67.102.3       FFFFFFFF  1       1566    9.67.102.3
 SPF   9.67.102.7       FFFFFFFF  2       1561    CTC7TO3
 DIR*  9.67.103.0       FFFFFF00  1       1575    9.67.103.7
 RIP   9.67.103.6       FFFFFFFF  1       30      9.67.103.6
 SPF   9.67.105.4       FFFFFFFF  2       1545    9.67.100.8
 SPF   9.67.105.8       FFFFFFFF  2       1561    9.67.106.4
 DIR*  9.67.106.0       FFFFFF00  1       1576    9.67.106.7
 SPF   9.67.106.4       FFFFFFFF  1       1566    9.67.106.4
 SPF   9.67.106.7       FFFFFFFF  2       1561    CTC7TO4
 DIR*  9.67.107.0       FFFFFF00  1       1577    9.67.107.7
 SPF   9.67.107.5       FFFFFFFF  1       1574    9.67.107.5
 SPF   9.67.107.7       FFFFFFFF  2       1566    CTC7TO5
 SPF   9.67.108.2       FFFFFFFF  2       1380    9.67.106.4
 SPF   9.67.108.4       FFFFFFFF  3       1380    9.67.106.4
SBNT   10.0.0.0         FF000000  1       1368    NONE
SPE2   10.1.1.0         FFFFFF00  0       1379    9.67.106.4
SPE2   10.1.1.1         FFFFFFFF  0       1379    9.67.106.4
SBNT   20.0.0.0         FF000000  1       1549    NONE
SPE2   20.1.1.0         FFFFFF00  0       1379    9.67.107.5
SPE2   20.1.1.1         FFFFFFFF  0       1379    9.67.107.5
 RIP   30.0.0.0         FF000000  2       30      9.67.103.6
 RIP   30.1.1.0         FFFFFF00  2       30      9.67.103.6
 RIP % 30.1.1.4         FFFFFFFF  2       30      9.67.103.6
 RIP % 30.1.1.8         FFFFFFFF  2       30      9.67.103.6
SPE2   130.200.0.0      FFFF0000  0       1379    9.67.100.8      (2)
```

```
SPE2   130.200.1.1      FFFFFFFF  0       1379    9.67.102.3
SPE2   130.200.1.18     FFFFFFFF  0       1379    9.67.100.8
SPE2   130.201.0.0      FFFF0000  0       1379    9.67.100.8       (2)
SPE2   130.202.0.0      FFFF0000  0       1379    9.67.100.8       (2)
                       0 NETS DELETED, 4 NETS INACTIVE
```

Displaying the routes to a specific destination: To display information about the routes to a specific destination, enter the following command:

```
D TCPIP,TCPCS7,OMP,RTTABLE,DEST=130.201.0.0
EZZ7874I ROUTE EXPANSION 798
DESTINATION:    130.201.0.0
MASK:           255.255.0.0
ROUTE TYPE:     SPE2
DISTANCE:       0
AGE:            1485
NEXT HOP(S):    9.67.100.8        (CTC7TO8)
                9.67.102.3        (CTC7TO3)
```

Displaying all of the RIP configuration information: To display all of the RIP configuration information, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,LIST,ALL
EZZ7843I RIP CONFIGURATION 800
TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
STACK AFFINITY: TCPCS7
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
CTC7TO6          9.67.103.7      RIP-2 MULTICAST.
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0

EZZ7844I RIP ROUTE ACCEPTANCE
ACCEPT RIP UPDATES ALWAYS FOR:
  30.1.1.8          30.1.1.4
```

Displaying information about configured RIP interfaces: To display information about configured RIP interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,LIST,IFS
EZZ7843I RIP CONFIGURATION 806
TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
STACK AFFINITY: TCPCS7
RIP: ENABLED
RIP DEFAULT ORIGINATION: ALWAYS, COST = 1
PER-INTERFACE ADDRESS FLAGS:
CTC7TO6          9.67.103.7      RIP-2 MULTICAST.
                                 SEND NET AND SUBNET ROUTES
                                 RECEIVE NO DYNAMIC HOST ROUTES
                                 RIP INTERFACE INPUT METRIC: 1
                                 RIP INTERFACE OUTPUT METRIC: 0
                                 RIP RECEIVE CONTROL: ANY
```

RIP RECEIVE CONTROL indicates what level of RIP updates can be received over the interface. Values are:

ANY RIP1 and RIP2 updates can be received

RIP1 Only RIP1 updates can be received.

RIP2 Only RIP2 updates can be received.

Displaying the routes to be unconditionally accepted: To display the routes to be unconditionally accepted, as configured with the Accept_RIP_Route statement, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,LIST,ACCEPTED
EZZ7844I RIP ROUTE ACCEPTANCE 808
ACCEPT RIP UPDATES ALWAYS FOR:
  30.1.1.8          30.1.1.4
```

Displaying current run-time information about RIP interfaces: To display current, run-time information about RIP interfaces, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,IF
EZZ7859I RIP INTERFACES 810
IFC ADDRESS      IFC NAME         SUBNET MASK     MTU   DESTINATION
9.67.103.7       CTC7TO6          255.255.255.0   1024  0.0.0.0
```

Displaying current run-time information about a specific RIP interface: To display current, run-time information about a specific RIP interface, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,IF,NAME=CTC7TO6
EZZ7860I RIP INTERFACE DETAILS 812
INTERFACE ADDRESS:      9.67.103.7
INTERFACE NAME:         CTC7TO6
SUBNET MASK:            255.255.255.0
MTU                     1024
DESTINATION ADDRESS:    0.0.0.0

RIP VERSION:            2     SEND POIS. REV. ROUTES: YES
IN METRIC:              1     OUT METRIC:             0
RECEIVE NET ROUTES:     YES   RECEIVE SUBNET ROUTES:  YES
RECEIVE HOST ROUTES:    NO    SEND DEFAULT ROUTES:    NO
SEND NET ROUTES:        YES   SEND SUBNET ROUTES:     YES
SEND STATIC ROUTES:     NO    SEND HOST ROUTES:       NO
RIP RECEIVE CONTROL:    ANY

SEND ONLY: ALL
```

RIP RECEIVE CONTROL indicates what level of RIP updates can be received over the interface. Values are:

ANY RIP1 and RIP2 updates can be received.

RIP1 Only RIP1 updates can be received.

RIP2 Only RIP2 updates can be received.

Displaying the global RIP filters: To display the global RIP filters, enter the following command:

```
D TCPIP,TCPCS7,OMP,RIP,FILTERS
EZZ8016I GLOBAL RIP FILTERS 814
SEND ONLY: ALL

FILTERS: NOSEND          10.1.1.0            255.255.255.0
EZZ8026I IGNORE RIP NEIGHBOR
                         9.67.103.9
                         9.67.103.10
```

Sample OMPROUTE configuration files

The following is an example of a pure OSPF environment (from TCPCS4 in the Figure 33 on page 158).

```
RouterID=4.4.4.4;
Area
     Area_Number = 0.0.0.0;
Area
```

```
     Area_Number = 1.1.1.1;
OSPF_Interface
     IP_Address=9.67.108.4
     Name = CTC4TO2
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=0.0.0.0
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.106.4
     Name = CTC4TO7
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.105.4
     Name = CTC4TO8
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.101.4
     Name = CTC4TO3
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=4.4.4.4
     Name = VIPA1A
     Subnet_Mask=255.255.255.252
     Attaches_To_Area=1.1.1.1
     Cost0 = 1;
Virtual_Link
     Virtual_Endpoint_RouterID=7.7.7.7
     Links_Transit_Area=1.1.1.1;
```

The following is an example of mixed OSPF and RIP environments (from TCPCS7 in Figure 33 on page 158).

```
;*********************************
;  OSPF Configuration Statements  *
;*********************************
RouterID=7.7.7.7;
Area
     Area_Number = 0.0.0.0;
Area
     Area_Number = 1.1.1.1;
AS_Boundary_Routing
     Import_Subnet_Routes=YES
     Import_RIP_Routes=YES;
OSPF_Interface
     IP_Address=9.67.107.7
     Name = CTC7TO5
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=0.0.0.0
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.106.7
     Name = CTC7TO4
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
```

```
OSPF_Interface
     IP_Address=9.67.102.7
     Name = CTC7TO3
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.100.7
     Name = CTC7TO8
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     MTU = 1024
     Cost0 = 1;
OSPF_Interface
     IP_Address=9.67.104.7
     Name = NBMA7
     Subnet_Mask=255.255.255.0
     Attaches_To_Area=1.1.1.1
     Non_Broadcast=YES
     NB_Poll_Interval=180
     MTU = 1024
     Cost0 = 1
     DR_Neighbor=9.67.104.15
     No_DR_Neighbor=9.67.104.16
     No_DR_Neighbor=9.67.104.25;
OSPF_Interface
     IP_Address=7.7.7.7
     Name = VIPA1A
     Subnet_Mask=255.255.255.252
     Attaches_To_Area=1.1.1.1
     Cost0 = 1;
Range
     IP_Address=9.67.101.0
     Subnet_Mask=255.255.255.0
     Area_Number=1.1.1.1
     Advertise=NO;
Virtual_Link
     Virtual_Endpoint_RouterID=4.4.4.4
     Links_Transit_Area=1.1.1.1;
;*********************************
;  RIP Configuration Statements  *
;*********************************
Originate_RIP_Default
     Condition=Always;
Accept_RIP_Route
     IP_Address=30.1.1.4;
Accept_RIP_Route
     IP_Address=30.1.1.8;
Filter=(nosend,10.1.1.0,255.255.255.0);
RIP_Interface
     IP_Address=9.67.103.7
     Name = CTC7TO6
     Subnet_Mask=255.255.255.0
     Receive_Dynamic_Hosts=NO
     MTU = 1024
     RipV2=YES;
```

The following is an example of a pure RIP environment (from TCPCS6 in Figure 33 on page 158).

```
RIP_Interface
     IP_Address=9.67.103.6
     Name = CTC6TO7
     Subnet_Mask=255.255.255.0
     MTU = 1024
     Send_Static_Routes=YES
```

```
        Send_Host_Routes=YES
        RipV2=YES;
    Interface
        IP_Address=6.6.6.6
        Name = VIPA1A
        Subnet_Mask=255.255.255.252;
```

IPv6 dynamic routing

Enabling IPv6 router discovery in z/OS Communications Server requires no additional z/OS Communications Server configuration. All that is needed is at least one IPv6 interface that is defined and started, and at least one adjacent router through that interface that is configured for IPv6 router discovery. If these things exist, then z/OS Communications Server begins receiving router advertisements from the adjacent routers. Depending on the configuration in the adjacent routers, the following types of routes may be learned from the received router advertisements:

- Default route for which the originator of the router advertisement is the next hop
- Direct routes (no next hop) to prefixes that reside on the link shared by z/OS Communications Server and the originator of the router advertisement.

Multiple default routes and multiple direct prefix routes to a single prefix may be learned through router advertisements. If an adjacent router resides on a link onto which z/OS Communications Server TCPIP has multiple IPv6 interfaces, there will be multiple routes to each route learned through the adjacent router's router advertisement (one route through each interface onto the link). Also, if default routes are learned from the router advertisements originated by multiple adjacent routers, there will be multiple default routes (one with each of these adjacent routers as next hop). When this condition of multiple routes exists, TCP/IP will use those routes according to the setting of the MULTIPATH parameter on the IPCONFIG6 statement.

If there are static non-replaceable routes to the destinations in the router advertisements, the dynamic routes will not be added to the stack routing table.

Verification of routing (Static and dynamic)

- If static routes are used, an indirect route must not be defined before the route to its first hop is defined. The following example shows an incorrect configuration.

```
BEGINRoutes ;first BEGINRoutes in the profile
;Network/mask           FirstHop       LinkName  PacketSize
Route 9.67.104.0/24     9.67.105.8     CTC4TO8   MTU 1500
Route 9.67.105.0/24     =              CTC4TO8   MTU 1500
Route FEC0:0:0:A1B::/64 FE80::1:2:3:3  OSAQDIO46 MTU 5000
Route FE80::1:2:3:3/128 =              OSAQDIO46 MTU 5000
ENDRoutes
```

  When configured incorrectly, the following error messages are displayed:

```
EZZ0657I ROUTE LIST ENTRY ON LINE 28 FOR DESTINATION 9.67.104.0 IS
UNREACHABLE THROUGH INTERFACE 9.67.105.8 ON CTC4TO8
EZZ0657I ROUTE LIST ENTRY ON LINE 30 FOR DESTINATION FEC0:0:0:A1B:: IS
UNREACHABLE THROUGH INTERFACE FE80::1:2:3:3 ON OSAQDIO46
```

- If OMPROUTE is used for the OSPF protocol only and AUTOLOG is not configured correctly (see "Autolog considerations for OMPROUTE" on page 172), OMPROUTE will be periodically restarted and the following messages are displayed:

```
$HASP100 OMPROUTE ON STCINRDR
$HASP373 OMPROUTE STARTED
IEF403I OMPROUT1 - STARTED
      OMPROUT1    OMPROUTE    BPXBATCH    0000
EZZ7800I OMPROUTE STARTING
EZZ7872I OMPROUTE FOUND ANOTHER ROUTING APPLICATION ALREADY ACTIVE
EZZ8074I OMPROUTE PROCESSING ERROR
EZZ7805I OMPROUTE EXITING ABNORMALLY - RC(11)
OMPROUT1    *OMVSEX     BPXPRECP    0011
IEF404I OMPROUT1 - ENDED
$HASP395 OMPROUT1 ENDED
```

- If a configuration statement in the OMPROUTE configuration file has a missing semicolon, the syntax checker might issue the following message:

  ```
  EZZ7830I SYNTAX ERROR AT LINE 22 OF OMPROUTE CONFIGURATION FILE
  PROCESSING END OF FILE
  ```

Verifying connections with NETSTAT, PING, and TRACERTE

The interfaces were verified with the instructions in Chapter 1, “Configuration overview” on page 3. The first thing to verify is that the devices and interfaces are started. In the case of point-to-point links like the CTCs in TCPCS4, the following message is written to the z/OS console when the device starts:

```
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE CTCE02
```

In the case of IPv6 interfaces like OSAQDIO46 in TCPCS4, the following message is written to the z/OS console when the interface starts:

```
EZZ4340I INITIALIZATION COMPLETE FOR INTERFACE OSAQDIO46
```

The same information can be determined from NETSTAT DEV. Following is a portion of the output of NETSTAT DEV with the CTCE02 device shown as ready. The NETSTAT DEV can be issued on TCPCS4 and TCPCS7 to verify that the devices on both systems are ready.

```
DEVNAME: CTCE02            DEVTYPE: CTC       DEVNUM: 0E00
  DEVSTATUS: READY
  LNKNAME: CTC4TO7           LNKTYPE: CTC        LNKSTATUS: READY
    NETNUM:  0   QUESIZE: 0
    BYTESIN: 488                     BYTESOUT: 1092
    ACTMTU:  32760
  BSD ROUTING PARAMETERS:
    MTU SIZE: 01500             METRIC: 01
    DESTADDR: 0.0.0.0           SUBNETMASK: 255.255.255.0
  MULTICAST SPECIFIC:
    MULTICAST CAPABILITY: YES
    GROUP             REFCNT
    -----             ------
    224.0.0.5         0000000001
    224.0.0.1         0000000001
```

Following is a portion of the output of NETSTAT DEV with an IPv6 interface (OSAQDIO46) shown as ready.

```
DEVNAME: OSAQDIO2          DEVTYPE: MPCIPA
  DEVSTATUS: READY
  INTFNAME: OSAQDIO46        INTFTYPE: IPAQENET6 INTFSTATUS: READY
    NETNUM:  0   QUESIZE: 0   SPEED: 0000001000
    BYTESIN: 592                     BYTESOUT: 1008
    MACADDRESS: 0002559A3F65
    DUPADDRDET: 1
    CFGROUTER: NON                   ACTROUTER: NON
    RTRHOPLIMIT: 5
    CFGMTU:  NONE                    ACTMTU:   8992
  MULTICAST SPECIFIC:
```

```
    MULTICAST CAPABILITY: YES
    REFCNT      GROUP
    ------      -----
    0000000001  FF02::1:FF03:1
    0000000001  FF02::1
```

If the devices do not have a LnkStatus or IntfStatus of Ready, this must be resolved before continuing. There are several things that might cause the LnkStatus or IntfStatus to not be ready. For example, the device might not be defined to z/OS correctly, the device might not be defined in PROFILE.TCPIP correctly, and so on.

You can PING each others hosts within the network to verify indirect routes exist.

```
ping 9.67.107.7
 CS V1R4: Pinging host 9.67.107.7
 Ping #1 response took 0.048 seconds.
 READY
ping fec0:0:0:a1b:2:559a:3f65:3
 CS V1R4: Pinging host fec0:0:0:a1b:2:559a:3f65:3
 Ping #1 response took 0.051 seconds.
 READY
```

Use TRACERTE to verify that the correct route is being taken for each indirectly attached host:

```
tracerte 9.67.107.5
 CS V1R4: Traceroute to 9.67.107.5 (9.67.107.5)
 1 9.67.106.7 (9.67.106.7)  40 ms  7 ms  6 ms
 2 9.67.107.5 (9.67.107.5)  9 ms  8 ms  9 ms
 READY
```

Following is an IPv6 example for indirectly attached hosts:

```
tracerte fec0:0:0:a1c:2:36a4:b39a:7
 CS V1R4: Traceroute to fec0:0:0:a1c:2:36a4:b39a:7
 at IPv6 address: fec0:0:0:a1c:2:36a4:b39a:7
 1 fe80::1:2:3:4
 (fe80::1:2:3:4) 13 ms 25 ms 40 ms
 2 fec0:0:0:a1c:2:36a4:b39a:7
 (fec0:0:0:a1c:2:36a4:b39a:7) 29 ms 263 ms 196 ms
```

Chapter 5. Virtual IP Addressing

This chapter contains information about the following topics:

- Terminology
- Introduction to VIPA
- Moving VIPA (Upon outage of TCP/IP)
- Static VIPAs, Dynamic VIPAs (DVIPAs), and Distributed Dynamic VIPAs
- Using static VIPAs
- Using Dynamic VIPAs (DVIPAs)
- Choosing which form of Dynamic VIPA to use
- Configuring Distributed DVIPAs — Sysplex Distributor
- Resolution of DVIPA conflicts
- Other considerations
- DVIPAs and routing protocols

Terminology

Virtual IP Address (VIPA)
A VIPA is a generic term that refers to an internet address on a z/OS host that is not associated with a physical adapter. There are two types of VIPAs:

- A *Static VIPA* cannot be changed except through a VARY TCPIP,,OBEYFILE operator command.
- A *Dynamic VIPA (DVIPA)* can move to other TCP/IP stack members in a sysplex or it can be activated by an application program or by a supplied utility. Dynamic VIPAs are used to implement Sysplex Distributor as described in "Considerations for VIPA" on page 65.

Distributed DVIPA
A distributed DVIPA, which is a special type of DVIPA, can distribute connections within a Sysplex.

Dynamic routing
VIPAs are designed to interoperate with a dynamic routing daemon. Therefore, it is highly recommended that a routing daemon be used on a z/OS host that uses VIPAs.

Introduction to VIPA

Traditionally, an IP address is associated with each end of a physical link (or each point of access to a shared-medium LAN), and the IP addresses are unique across the entire visible network, which can be the Internet or a closed intranet. The majority of IP hosts have a single point of attachment to the network, but some hosts (particularly large server hosts) have more than one link into the network. A TCP/IP host with multiple points of attachment also has multiple IP addresses, one for each link.

Within the IP routing network, failure of any intermediate link or adapter disrupts end user service only if there is not an alternate path through the routing network. Routers can route IP traffic around failures of intermediate links in such a way that the failures are not visible to the end applications or IP hosts. However, because an IP packet is routed based on ultimate destination IP address, if the adapter or link

associated with the destination IP address fails, there is no way for the IP routing network to provide an alternate path to the stack and application. Endpoint (source or destination) IP adapters and links thus constitute single points of failure. While this might be acceptable for a client host, where only a single user will be cut off from service, a server IP link might serve hundreds or thousands of clients, all of whose services would be disrupted by a failure of the server link.

The Virtual IP Address (VIPA) removes the adapter as a single point of failure by providing an IP address that is associated with a stack without associating it with a specific physical network attachment. Because the virtual device exists only in software, it is always active and never experiences a physical failure. A VIPA has no single physical network attachment associated with it. Also, the TCP/IP stack does not maintain interface counters for VIPA interfaces (VIRTUAL links).

To the routing network, a VIPA appears to be a host address indirectly attached to the z/OS. When a packet with a VIPA destination reaches the stack, the IP layer recognizes the address and passes it to the protocol layer in the stack.

The failure of the physical interface can be extended to the failure of the TCP/IP address space, the entire z/OS, or for planned outages. A VIPA just needs to move to a backup stack, and the routes to the VIPA need to be updated. Then clients can transparently connect to the backup stack. This process is known as VIPA Takeover.

VIPA Takeover improved with the introduction of Dynamic Virtual IP Address (DVIPA) and Distributed Dynamic Virtual IP Address (Distributed DVIPA). The DVIPA function improves VIPA Takeover by allowing a system programmer to plan for system outages and provide for backup systems to take over without operator intervention or external automation. The Distributed DVIPA function allows the connections for a single DVIPA to be serviced by applications on several stacks listed in the configuration statement (the distribution list). This adds the benefit of limiting the scope of an application or stack failure, while also providing enhanced work load balancing.

In general, z/OS configured with VIPA provides the following advantages:

- Automatic and transparent recovery from device and adapter failure.

  When a device (for example, 3172, or channel-attached 2216) or adapter (for example, a Token Ring or FDDI card) fails, if there is another device or link that provides the alternate paths to the destination:

  – IP will detect the failure, find an alternate path for each network, and route outbound traffic to hosts and routers on those networks via alternate paths.
  – Inbound and outbound traffic will not need to reestablish the active TCP connections that were using the failed device.

- Recovery from z/OS TCP/IP stack failure (where an alternate z/OS TCP/IP stack has the necessary redundancy).

  Assuming that an alternate stack is installed to serve as a backup, the use of VIPAs enables the backup stack to activate the VIPA address of the failed stack.

  Connections on the failed primary stack will be disrupted but they can be reestablished on the backup using the same IP as the destination. In addition, the temporarily reassigned VIPA address can be restored to the primary stack after the cause of failure has been removed.

  Note: For connection requests originating at a z/OS TCP/IP stack, tolerance of device and adapter failures can be achieved by using the SOURCEVIPA option. For IPv6 connection requests to have the same tolerance, the IPv6

SOURCEVIPA configuration option must be enabled and a VIPA interface must be specified with the SOURCEVIPAINT keyword on the INTERFACE statement associated with the failed device or adapter.

With this option, static VIPA addresses are used as the source IP addresses in outbound datagrams for TCP, RAW, UDP (except routing protocols), and ICMP requests.

- Limited scope of a stack or application failure.

  If a DVIPA is distributed among several stacks, the failure of only one stack affects only the subset of clients connected to that stack. If the distributing stack experiences the failure, a backup assumes control of the distribution and maintains all existing connections.
- Enhanced workload management through distribution of connection requests.

  With a single DVIPA being serviced by multiple stacks, connection requests and associated workloads can be spread across multiple z/OS images according to Workload Manager (WLM) and Service Level Agreement policies (for example, QOS).
- Allows the non-disruptive movement of an application server to another stack so that workload can be drained from a system in preparation for a planned outage.

Moving a VIPA (For TCP/IP outage)

While a VIPA provides non-disruptive rerouting of IP data during failure of a physical interface, termination of the stack or the associated z/OS (including planned outages) will disrupt connections or UDP sessions to applications on the terminated stack. While failure of the TCP connection or UDP session will be visible to the clients, the duration of the outage is determined by how long the client application is unable to reconnect to an appropriate server application. Because it is common in large enterprises to have multiple instances of an application residing on different z/OS images, if the VIPA address can be moved to another stack that supports the application, the clients can reconnect and the perceived outage will be over.

An IP address associated with a particular physical device is unavailable until the owning stack is restarted; however, a VIPA is not associated with any particular physical interface. If termination of a stack is detected and a suitable application already is active on another stack, the VIPA can be moved. Connections on the terminated stack will be disrupted, but they can be reestablished on the backup stack using the original VIPA.

Movement of a static VIPA to a backup stack can be accomplished by using VARY TCPIP,,OBEYFILE commands on the backup. The OBEYFILE data set must contain an appropriate set of DEVICE, LINK, HOME, and optionally, BSDROUTINGPARM statements for IPv4 static VIPAs or INTERFACE statements for IPv6 static VIPAs. If OMPROUTE is used as the routing daemon, an appropriate interface statement is needed in the OMPROUTE configuration file. If the TCP/IP configuration file with the statements defining the VIPA is created in advance, the transfer can be accomplished via automation. This procedure is documented in *z/OS Communications Server: IP Configuration Reference*. Movement of a DVIPA, on the other hand, can be accomplished by configuring a stack to backup a specific DVIPA that is defined on another stack. In this case, failure of the defining stack causes the DVIPA to move without operator intervention or extra automation. See “Planning for Dynamic VIPA Takeover” on page 216 for more information. Regardless of the type of VIPA to be moved, it is up to the system programmer or operator to ensure that the VIPA is moved to a backup stack that has the appropriate server applications.

In the absence of a failure, a VIPA is just like any other IP address, and routing for a VIPA is the same as for an IP address associated with a physical link.

Static VIPAs, Dynamic VIPAs (DVIPAs), Distributed DVIPAs

z/OS TCP/IP stack supports two types of VIPAs: static and dynamic. Dynamic VIPAs (DVIPAs) can be used to distribute connections in a sysplex. This is referred to as a Distributed DVIPA.

All three VIPAs can coexist on a given stack, but there are differences in how these VIPAs are configured and used.

Static VIPAs have the following characteristics:

- They can be activated during TCP/IP initialization or VARY TCPIP,,OBEYFILE processing, and are configured using an appropriate set of DEVICE, LINK, HOME, and optionally, OMPROUTE configuration statements or BSDROUTINGPARMS statements for IPv4 Static VIPAs or INTERFACE statements for IPv6 Static VIPAs.
- Using the SOURCEVIPA configuration option, static VIPAs can be used as the source IP address for outbound datagrams for TCP, RAW, UDP (except routing protocols), and ICMP requests. For IPv6 static VIPAs to be used as source addresses, the SOURCEVIPA configuration option must be enabled and the VIPA interface must appear on the SOURCEVIPAINT keyword on some other INTERFACE statement. This provides tolerance of device and adapter failures for connection requests originating at a z/OS TCP/IP stack.
- They can be moved to a backup stack after the original owning stack has failed, by using VARY TCPIP,,OBEYFILE processing to configure the VIPA on the backup stack and updating the routers.
- The number of static VIPAs on a stack is limited only by the range of host IP addresses that are available for that host.

Dynamic VIPAs have the following characteristics:

- They can be configured to be moved dynamically from a failing stack to a backup stack within the same sysplex without operator intervention or external automation.
- They can be dynamically activated by an application program.
- They can distribute connections within a sysplex.
- They can be specified on a TCPSTACKSOURCEVIPA statement. This allows a user to specify one Dynamic VIPA to be used as the source IP address for outbound datagrams for TCP-only requests.
- Unlike static VIPAs, Dynamic VIPAs:
  - Are limited to 256 per stack.
  - Cannot be specified as the VIPA used by Enterprise Extender for connectivity purposes. (See “Configuring static VIPAs for Enterprise Extender” on page 214 for details.)

Distributed DVIPAs have the following characteristics:

- Have all the characteristics of DVIPAs, but cannot be dynamically activated by an application program.
- One stack defines a DVIPA and advertises its existence to the network. Stacks in the target distribution list activate the DVIPA and accept connection requests.
- Connection workload can be spread across several stacks.

See “Configuring Distributed DVIPAs — Sysplex Distributor” on page 224 for more detailed descriptions.

Using static VIPAs

The following sections describe how to configure static VIPAs, the special case of static VIPAs and Enterprise Extender, and how to implement static VIPA Takeover.

Because a VIPA is associated with a z/OS TCP/IP stack and it is not associated with a specific physical network attachment, it can be moved to a stack on any image in the sysplex or *even* to any z/OS TCP/IP stack if the address fits into the network configuration.

Configuring static VIPAs for a z/OS TCP/IP stack

To configure a static VIPA address in one stack, follow these steps:

1. When configuring static VIPAs for the IPv4 network, add VIPA DEVICE, LINK, HOME, and optionally, BSDROUTINGPARMS statements for each static VIPA to be defined. When configuring static VIPAs for the IPv6 network, add INTERFACE statements of type VIRTUAL6 for each static VIPA to be defined.

   Note: A VIPA link or VIPA interface cannot be coded on a static route in the GATEWAY or BEGINROUTES statements.
2. For IPv4 networks, if tolerance of device and adapter failures is desired for connection requests originating at a z/OS TCP/IP stack, specify the SOURCEVIPA option in the IPCONFIG statement. For this option to work properly, the receiving nodes in the network must be configured to recognize the SOURCEVIPA addresses using the static or dynamic routing protocols. Otherwise, timeouts for the connection or request responses will occur as a result of the VIPA addresses being network unreachable. If TCPSTACKSOURCEVIPA is specified, it overrides SOURCEVIPA for outbound IPv4 TCP connections. For more information on configuring IPv4 SOURCEVIPA or TCPSTACKSOURCEVIPA addresses, refer to *z/OS Communications Server: IP Configuration Reference*.
3. For IPv6 networks, if tolerance of device and adapter failures is desired for connection requests originating at a z/OS TCP/IP stack, specify the SOURCEVIPA option in the IPCONFIG6 statement and specify a VIPA interface with the SOURCEVIPAINTerface keyword on the INTERFACE statement of the real (physical) interface. For more information on configuring IPv6 SOURCEVIPA addresses, refer to *z/OS Communications Server: IP Configuration Reference*.
4. For host name resolution of a VIPA address, configure the domain name servers to associate the host name with the VIPA.
5. Configure the routing daemon to advertise the presence of the VIPA (IPv4 only; dynamic routing protocols for IPv6 are not supported).

Figure 34 on page 214 illustrates a simple configuration showing multiple network attachments using a single static VIPA address. Since any other network interface can be used with static VIPA’s, refer to “Setting up physical characteristics in PROFILE.TCPIP” on page 115 for descriptions of other network interfaces. The simple configuration will be used as the TCPCS6 system throughout this chapter.

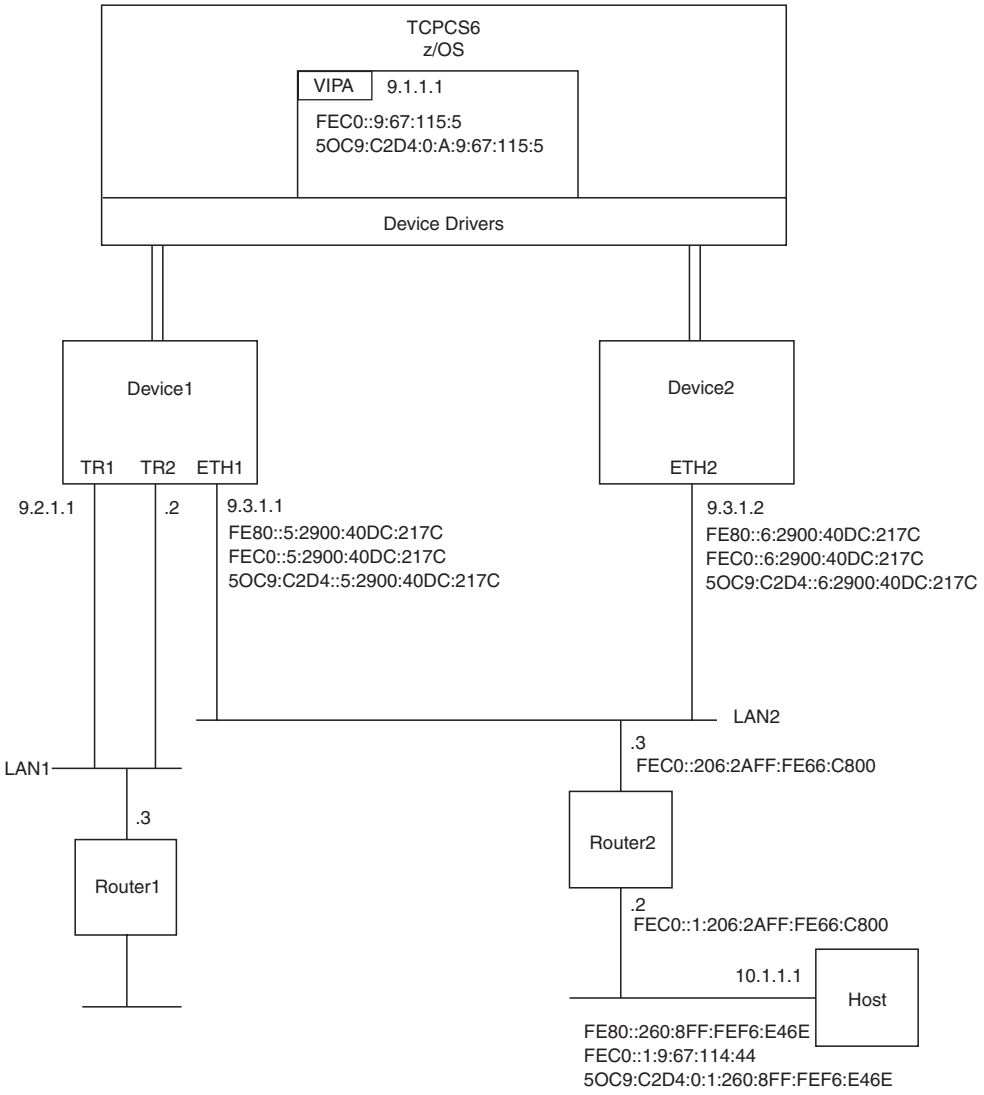


Figure 34. Static VIPA configuration

Configuring static VIPAs for Enterprise Extender

Defining at least one static VIPA is required by VTAM to access the IP network. Since VTAM does not move within a sysplex, a dynamic VIPA cannot be used. VTAM will use the VIPA address specified on the VTAM IPADDR start option. If the option is not used, VTAM will use the first static VIPA in the HOME list. If remote APPN nodes use a host name and not a host address to define the destination of an Enterprise Extender connection, the domain name server must return the VIPA address used by VTAM for the host name.

For more information about Enterprise Extender, refer to the following:

- *z/OS Communications Server: SNA Network Implementation Guide*
- http://www.ibm.com/software/network/library/whitepapers/eextender.html
- http://www.ibm.com/software/network/library/whitepapers/eemsthtm/eemst.htm
- IBM Redbook, *SNA and TCP/IP Integration (SG24–5291–00)*

Note: This document is also available at http://www.redbooks.ibm.com.

Considerations when using static VIPAs with IPv6

When static VIPAs are configured for use with IPv6, it is recommended that the prefixes of the IPv6 VIPA addresses be different than the prefixes used for addresses assigned to real interfaces. This reduces the likelihood of address collisions between the manually configured VIPA addresses and the autoconfigured addresses of the real interfaces.

To allow other hosts that share links with the z/OS TCP/IP stack to access the IPv6 VIPA addresses, without the need for manual route configuration, a router on each of the links should include the VIPA prefix in its router advertisements. The router advertisements should define the prefix as being on-link and should indicate that the prefix should not be used for autoconfiguration.

Planning for static VIPA Takeover and Takeback

Because a VIPA is associated with a z/OS TCP/IP stack and is not associated with a specific physical network attachment, it can be moved to a stack on any image in the sysplex or even to any z/OS TCP/IP stack as long as the address fits into the network configuration. Moving a static VIPA can be done manually by an operator or by customer-programmed automation. Movement of the static VIPA allows other hosts that were connected to the primary stack to reestablish connections with a backup TCP/IP stack using the same VIPA. After the primary TCP/IP stack has been restored, the reassigned VIPA address can be moved back.

Consider the following when backing up and restoring a z/OS TCP/IP stack:

- All connections on the failing host will be disrupted.
- The client can use any ephemeral port number when reestablishing the connection to the backup server.
- Having a different port number for the backup and primary server is not recommended. For example, if the primary FTP used port 21 and the backup FTP used port 1021, when backing up and restoring a z/OS TCP/IP stack, the client would have to know whether to use port 21 or 1021.

Using Dynamic VIPAs (DVIPAs)

DVIPA support allows:

- Dynamic movement of a VIPA from a failing TCP/IP stack to a backup stack
- Dynamic allocation of a VIPA by an application program

Dynamic VIPAs (DVIPAs) are IP addresses like all other IP addresses associated with a TCP/IP, and they appear as though they had been defined at the end of the HOME list.

Configuring Dynamic VIPA (DVIPA) support

Unlike static VIPAs, DVIPAs are not configured using DEVICE, LINK, and HOME statements. The configuration statements for the DVIPA support are contained within the VIPADYNAMIC and ENDVIPADYNAMIC block and consist of the following:

- VIPADEFINE and VIPABACKUP statements used to configure DVIPAs to be dynamically moved from a failing TCP/IP to a backup TCP/IP
- VIPARANGE used to specify a range of IP addresses which may be dynamically activated as a VIPA by an application program

- VIPADELETE used to delete existing DVIPAs
- VIPADISTRIBUTE used to configure a DVIPA as a distributed DVIPA and designate the target stacks

The following sections discuss how these statements are used to provide the DVIPA support. For syntax details, see *z/OS Communications Server: IP Configuration Reference*.

When Dynamic VIPAs (DVIPAs) are used for VIPA Takeover together with DNS/WLM in a sysplex, code all of the DVIPAs in the sysplex under each host name in the DNS forward domain data file for the cluster zone. This will circumvent manual intervention in the DNS data files when a DVIPA is taken over or given back and will not cause any undesirable effects in DNS/WLM function.

Planning for Dynamic VIPA Takeover

Movement by network management automation or operator intervention is not always desirable. Operator intervention takes time and is subject to errors. Automation requires proper detection of the failure and is also prone to error if the failure does not produce the exact console messages anticipated.

Dynamic VIPA Takeover function addresses this problem. It is important to understand that Dynamic VIPA Takeover does not introduce functions that could not be accomplished by operator action or automation. It just removes the dependency on human detection of the error or customer programming for automation. Dynamic VIPA Takeover is completely accomplished by the TCP/IP stacks.

DVIPA Takeover is possible when a DVIPA is configured as active (via VIPADEFINE) on one stack and as backup (via VIPABACKUP) on another stack within the sysplex. When the stack on which the DVIPA is active terminates, then the backup stack will automatically activate the DVIPA and notify the routing daemon. For DVIPA Takeover to be useful, the applications that service the DVIPA addresses must be available on the backup stacks. In the absence of the application, the DVIPA will be active, but client connections to the application will still fail.

A determination of how the workload will be distributed among the backup stacks when the primary stack fails should be made. It is possible to designate a single stack as a backup and move all the workload to it, or the workload can be spread among several stacks. In the first case, only one DVIPA must be configured with a VIPADEFINE statement on the primary stack, and only one VIPABACKUP statement is required on the backup stack. The second option requires the definition of a VIPABACKUP statement for each stack that will assume responsibility for a subset of the primary's workload.

After determining the workload distribution, each of the secondary stacks will require a VIPABACKUP statement for the DVIPA it will be supporting.

The following example shows how to implement a single stack backup for multiple applications.

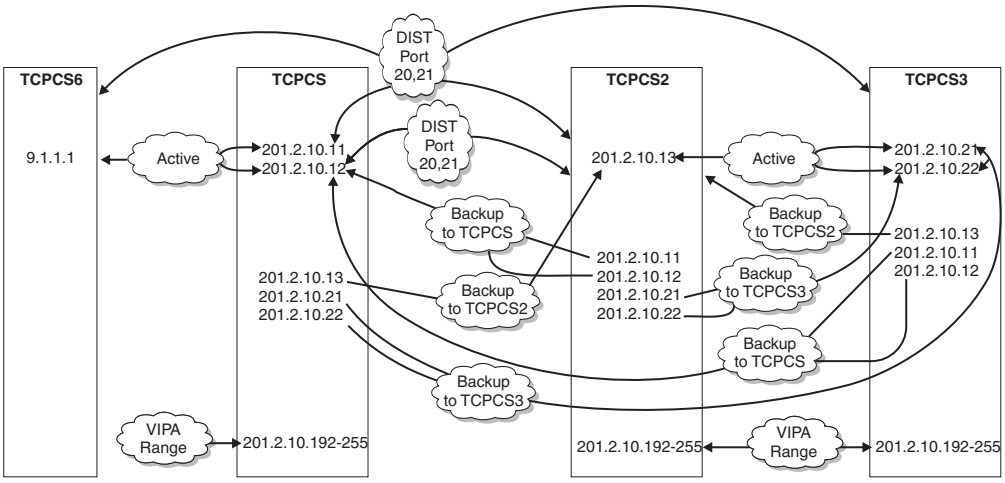


Figure 35. Sample DVIPA addressing in a sysplex environment

Stack TCPCS:

```
Uses VIPADEFINE to define 201.2.10.11
Has a Web server running that binds to INADDR_ANY.
  Web client programs use 201.2.10.11 as their destination address.
Has an FTP server running that binds to INADDR_ANY.
  FTP client programs use 201.2.10.11 as their destination address.
```

Stack TCPCS3:

```
Uses VIPABACKUP to define 201.2.10.11 as backup for stack TCPCS.
Has a Web server running that binds to INADDR_ANY.
Has an FTP server running that binds to INADDR_ANY.
```

In the preceding scenario, when stack TCPCS goes down, stack TCPCS3 receives all new connection requests for both the Web and FTP servers. FTP and Web client programs continue to use 201.2.10.11 as their destination address, but they now connect to stack TCPCS3.

The following example shows how to implement a multiple stack backup for multiple applications.

Stack TCPCS:

```
Uses VIPADEFINE to define 201.2.10.11 and 201.2.10.12
Has a Web server running that binds to INADDR_ANY.
  Web client programs use 201.2.10.11 as their destination address.
Has an FTP server running that binds to INADDR_ANY.
  FTP client programs use 201.2.10.12 as their destination address.
```

Stack TCPCS2:

```
Uses VIPABACKUP to define 201.2.10.11 as backup for stack TCPCS.
Has a Web server running that binds to INADDR_ANY.
```

Stack TCPCS3:

```
Uses VIPABACKUP to define 201.2.10.12 as backup for stack TCPCS.
Has an FTP server running that binds to INADDR_ANY.
```

In the preceding scenario, when stack TCPCS goes down, new connections for the Web server at 201.2.10.11 will connect with stack TCPCS2, and new connections for the FTP server at 201.2.10.12 will connect with stack TCPCS3.

Different application uses of IP addresses and DVIPAs

Not all IP-based server applications relate to IP addresses in the same way. Automated movement of DVIPAs, and the planning for dynamic VIPA Takeover, must take this difference into account.

Some applications will accept client requests on any IP address by binding to INADDR_ANY (for example, TN3270 or Web servers). The distinguishing feature of such an application is the function it provides (the particular set of SNA applications for TN3270 or the particular web pages for a Web server). If the function is replicated across multiple z/OS images in the sysplex, as is often the case for distributed workload, the DVIPA must merely be moved to a stack supporting the application. This scenario is called the Multiple Application-Instance Scenario. For the Multiple Application-Instance Scenario, the stacks in the sysplex do all the work of activating a DVIPA in the event of a failure.

For other types of applications, each application instance must have a unique IP address for one of the following reasons:

- The application instance cannot bind to INADDR_ANY.
- Clients might establish a relationship to that server application instance that can span multiple TCP connections, and the client must get connected back to the same server application instance while the relationship lasts.

This scenario is called the Unique Application-Instance Scenario and uses DVIPAs that are activated with an ioctl or a bind().

To maintain the relationship between an application instance and its DVIPA, the application must indicate to the stack that the DVIPA needs to be activated. This occurs in the following cases:

- When the application instance issues a bind() function call and specifies an IP address that is not active on the stack. The stack will activate the address as a DVIPA, provided it meets certain criteria. When the application instance closes the socket, the DVIPA is deleted.
- Some applications cannot be configured to issue bind() for a specific IP address, but are Unique Application-Instance Scenario applications. For such applications, a utility is provided (MODDVIPA), which issues SIOCSVIPA ioctl() to activate or deactivate the DVIPA. This utility can be included in a JCL procedure or OMVS script to activate the DVIPA before initiating the application. As long as the same JCL package or script is used to restart the application instance on another node in the event of a failure, the same DVIPA will be activated on the new stack. For information about the authorization required to execute the MODDVIPA utility, see “Using the MODDVIPA utility” on page 222.

Configuring Dynamic VIPAs

To allow continued and unchanged operation of static VIPAs in z/OS TCP/IP, DVIPAs are defined with configuration statements in the PROFILE.TCPIP data set. An overview of the relevant configuration statements is provided in the following sections, and also see “Verifying the DVIPAs in a sysplex” on page 241 for a description of the configuration statements. For an example of the VIPADYNAMIC/ENDVIPADYNAMIC configuration statements and display commands for Dynamic VIPA, see *z/OS Communications Server: IP Configuration Reference*.

Configuring the Multiple Application-Instance Scenario

For the Multiple Application-Instance Scenario, each instance is assigned a unique DVIPA. The VIPADEFINE keyword of the VIPADYNAMIC configuration statement is used to create the DVIPA on the stack where the DVIPA is normally expected to be active. When the VIPADEFINE statement is processed in a TCP/IP profile, corresponding DEVICE, LINK, HOME, and BSDROUTINGPARMS statements are generated automatically. Routing daemons are automatically informed.

Additional configuration is required on other stacks in the sysplex to indicate which stack should take over the DVIPA in the event of a failure. The VIPADYNAMIC statement has a VIPABACKUP keyword for this purpose. A VIPABACKUP configures the DVIPA but does not activate it until it is necessary. Because more than one TCP/IP can backup a single DVIPA, a rank parameter on the VIPABACKUP statement determines the order in which several stacks will assume responsibility for a DVIPA.

The stacks in the sysplex exchange information on all VIPADEFINEs and VIPABACKUPs defined in the sysplex, so that all are aware of which stack should take over a particular DVIPA. The list of backup stacks for a specific DVIPA can be different from the list of backup stacks for all other DVIPAs.

In the Multiple Application-Instance Scenario, instances of the application in question are activated among sysplex nodes according to some plan, presumably related to balancing workload across available capacity. This activation is done independently of VIPA Takeover. Configure the associated DVIPAs as follows:

1. For each instance of a particular application to be supported via DVIPA, add a VIPADEFINE statement to the TCP/IP profile for the TCP/IP associated with the application instance.
2. For each of the Dynamic VIPAs in Step 1, determine which application instance or instances should take over the workload (considering probable capacity and any other application-related considerations). If more than one TCP/IP is to provide backup for a DVIPA, determine the order in which the selected TCP/IPs should be designated as backup. Add a VIPABACKUP statement to each TCP/IP that is to provide backup for the DVIPA, with appropriate rank values to determine the order. Do this for each of the DVIPAs in Step 1.
3. Perform steps 1 and 2 for each other application to be supported by DVIPAs.

   Note: It is possible to share a Dynamic VIPA among several different applications, but in doing so, ensure that instances of all such applications will exist together on any TCP/IP to which the DVIPA may be moved in case of a failure.

After these steps are complete, start the affected TCP/IPs (or modify their configuration using VARY TCPIP,,OBEYFILE), if applicable, configure DNS for the application names, and start the application instances. From that point on, the TCP/IPs in the sysplex will collaborate to ensure that each Dynamic VIPA is kept active somewhere within the sysplex as long as there is at least one functioning TCP/IP which has been designated as backup for the Dynamic VIPA.

Configuring the Unique Application-Instance Scenario

The Unique Application-Instance Scenario ties a DVIPA to a specific instance of an application. To isolate errors in configuring applications, TCP/IP needs a mechanism to identify permissible DVIPAs. This is provided with one or more VIPARANGE statements. The VIPARANGE statement identifies a range of IP addresses which

can be activated as DVIPAs by an application instance. The VIPARANGE statement consists of a subnet mask and an IP address and thus defines a subnet for DVIPAs. More than one VIPARANGE statement with different ranges can be defined on a TCP/IP. VIPARANGE does not itself cause any DVIPAs to be activated. Rather, DVIPAs are activated either by an application issuing a bind() for a specific IP address, by use of the SIOCSVIPA ioctl() command issued by an authorized application, or by the MODDVIPA utility.

When an application issues bind() for a specific IP address or an address was selected using the BIND keyword on the PORT statement, the receiving stack checks it against addresses in the HOME list. If the IP address has already been activated on this stack (whether for a physical device, a static VIPA, or a Dynamic VIPA), the bind() execution is successful. If the IP address is not active on this TCP/IP, the current VIPARANGEs are checked to see if the IP address falls within one of them. If an appropriate VIPARANGE is found, it is activated as a DVIPA and the operation succeeds. If no appropriate VIPARANGE is found, or if the IP address is active elsewhere in the sysplex other than by a NONDISRUPTIVE bind(), the request fails and bind() returns EADDRNOTAVAIL.

When an authorized application issues the SIOCSVIPA ioctl() command to create a DVIPA, or when the MODDVIPA utility is executed in JCL or an OMVS script to activate a DVIPA on behalf of an application instance, the current VIPARANGES are checked to see whether the IP address falls within one of them. If an appropriate VIPARANGE is found, and the IP address is not currently active on this TCP/IP or elsewhere in the sysplex as an IP address or a VIPADEFINE/VIPABACKUP Dynamic VIPA, then the IP address is activated as a DVIPA. However if no appropriate VIPARANGE is found on this TCP/IP, or if the IP address is currently defined on this TCP/IP or configured elsewhere in the sysplex as an IP address or a VIPADEFINE/VIPABACKUP Dynamic VIPA, then the request fails with errno and errnojr set to indicate the reason for the failure and the utility ends with a nonzero condition code. See “Dynamic VIPA creation results” on page 236 for more information.

Note: If the requested IP address has been activated as a Dynamic VIPA by a bind() or SIOCSVIPA ioctl elsewhere in the sysplex, the result depends on how the stacks were configured. See “Dynamic VIPA creation results” on page 236 for more information.

In the Unique Application-Instance Scenario, each application instance is assigned a unique IP address as its DVIPA. Before defining individual DVIPAs, one or more blocks of IP addresses must be defined for these DVIPAs, and the individual DVIPAs must be defined from within the blocks. Each block should be represented as a subnet, so that a VIPARANGE statement can be defined for it.

Follow these steps when setting up any unique application instances:

1. For each application instance, assign a DVIPA from one of the blocks of IP addresses for this purpose. Do not assign an IP address which is also assigned to another application instance, or which is defined by VIPADEFINE for the Multiple Application-Instance Scenario. Configure the application to use this DVIPA (if it issues bind() for a particular IP address), or add the MODDVIPA utility to the JCL or OMVS script and configure the MODDVIPA utility to activate the DVIPA before starting the application, and to delete the DVIPA when the application ends.
2. For each application instance, determine on which stack the application instance will normally be executed and to which stacks the application instance could be

moved in case of failure of the normal stack or the application itself. For each such stack, add a valid VIPARANGE statement to the profile.

Note: The dynamic VIPA must be within the VIPARANGE subnet. The broadcast address and the net prefix cannot be used.

3. Perform steps 1 and 2 until all application instances have been allocated a unique DVIPA.

The application restart strategy should ensure that the worst-case failure scenario does not attempt to activate more than 256 DVIPAs on a single stack. If such an attempt is made, activation of the 257th DVIPA will fail, with possible resulting loss of connectivity from clients to the server application.

Note: The limit of 256 DVIPAs on a single TCP/IP applies to all DVIPAs, whether defined by VIPADEFINE/ VIPABACKUP configuration statements, through a VIPADISTRIBUTE statement on another stack, by a bind() call, or by executing the MODDVIPA utility.

Defining a single block makes the definition process easier, but also provides less individual control. Alternatively, since the smallest subnet consists of four IP addresses, defining a unique subnet for each DVIPA in this scenario *wastes* three other IP addresses that could have been used for DVIPAs.

Using the 'SIOCSVIPA' ioctl command

An ioctl command 'SIOCSVIPA' allows an application to create or delete a Dynamic VIPA on the stack where the application is running. The application issuing the 'SIOCSVIPA' ioctl command must be APF authorized and be running under a user ID with SuperUser authority. If the new profile for the MODDVIPA program has been defined under RACF, any user ID can be allowed to issue the SIOCSVIPA ioctl simply by being permitted to use this profile. For more information, see "Defining a RACF profile for MODDVIPA" on page 223.

To create a new Dynamic VIPA, the requested IP address must be within a subnet that has been previously specified by a VIPARANGE configuration statement in the PROFILE.TCPIP data set for this stack. The 'SIOCSVIPA' ioctl command can be used to delete any existing Dynamic VIPA on the stack, except for distributed DVIPAs.

The following example shows how to set up the 'SIOCSVIPA' ioctl command.

```
#include "ezbzdvpc.h"               /* header that contains
                                      the structure for
                                      'SIOCSVIPA' ioctl
                                       and needed constants*/
struct dvreq dv;                    /* the structure passed
                                         on the ioctl command*/
dv.dvr_version = DVR_VER1;           /*version               */
dv.dvr_length = sizeof(struct dvreq);  /* structure length     */
dv.dvr_option = DVR_DEFINE;           /* to define a new
                                         Dynamic VIPA.  Use
                                         DVR_DELETE to delete
                                         a dynamic VIPA        */
dv.dvr_addr.s_addr = inet_addr(my_ipaddr);  /* where my_ipaddr is
                                               a character string
                                               in standard
                                               dotted-decimal
                                               notation */
rc = ioctl(s, SIOCSVIPA, &dv);
```

The 'SIOCSVIPA' ioctl command sets nonzero errno and errnojr values to indicate error conditions. Refer to *z/OS Communications Server: IP and SNA Codes* for a description of the errnojr values returned.

Using the MODDVIPA utility

You can use the MODDVIPA utility to activate or delete a Dynamic VIPA. The utility can be initiated from JCL or an OMVS script. MODDVIPA must be loaded from an APF authorized library and be executed under a user ID with SuperUser authority. The user ID must also have an OMVS segment defined (or defaulted). If the new profile for the MODDVIPA program has been defined under RACF, any user ID can execute the MODDVIPA program simply by being permitted to use this profile. For more information, see "Defining a RACF profile for MODDVIPA" on page 223.

Note: In V2R8, this utility was called EZBXFDVP. The EZBXFDVP name will continue to work as it did in V2R8, but MODDVIPA is the preferred name and will be used throughout this document.

Input parameters: The input parameters for the utility are:

-p <tcpipname>
Specifies the TCP/IP which is to create or delete a DVIPA.

-c <IPaddress> or -d <IPaddress>
Specifies to create (-c) or delete (-d) the address (IP address) specified.

Notes:

1. The input parameters -p, -c, and -d must be entered in lowercase.
2. <*tcpipname*> must be entered in upper case.
3. <*IPaddress*> is dotted-decimal notation.
4. To create a DVIPA, the specified IP address must be within a subnet that has been previously specified by a VIPARANGE configuration statement in the PROFILE.TCPIP data set for the specified TCP/IP.

Output: The MODDVIPA utility sets the following exit (completion) codes for create (-c):

0 Success: The DVIPA was activated.

4 Warning: The requested DVIPA was not activated because the specified IP address is already active on this stack.

8 Error: The IP address was not defined as a DVIPA on this TCP/IP.

12 An error was found in the input parameters

The MODDVIPA utility sets the following exit (completion) codes for delete (-d):

0 Success: The Dynamic VIPA was deleted.

8 Error: The requested DVIPA was not deleted.

12 An error was found in the input parameters

Notes:

1. When an error is detected, the ernno text and errnojr value are printed to stderr.
2. If the IP address requested for the DVIPA is not within a VIPARANGE configured on this stack, completion code 8 is returned even if the IP address is currently active on this stack

Examples

Within JCL:

```
//TCPDVP EXEC  PGM=MODDVIPA,REGION=0K,TIME=1440,                        X
//             PARM='POSIX(ON) ALL31(ON)/-p TCPCS3 -c 1.2.3.4'
```

From OMVS shell:

```
moddvipa -p TCPCS3 -c 1.2.3.4
```

Defining a RACF profile for MODDVIPA

You can restrict access to the MODDVIPA (EZBXFDVP) program by defining a RACF profile under the SERVAUTH class and specifying the user IDs that are authorized to execute the SIOCSVIPA ioctl or the MODDVIPA utility program. You can decide on the level of control that is appropriate for your installation.

To restrict access to the SIOCSVIPA ioctl (and thus the MODDVIPA utility), you can define a RACF profile using the following example:

```
RDEFINE SERVAUTH (EZB.MODDVIPA.system_name.tcpip_name)
        UACC(NONE)

PERMIT EZB.MODDVIPA.system_name.tcpip_name
       ACCESS(READ) CLASS(SERVAUTH) ID(USER1)
```

where *system_name* is the name of the MVS system where the ID will execute the MODDVIPA utility or issue the SIOCSVIPA ioctl, and *tcpip_name* is the jobname of the TCP/IP started task. The jobname for started tasks, such as TCP/IP, is derived depending on how it is started:

- If the START command is issued with the name of a member in a cataloged procedure library (for example, S TCPIPX), the jobname will be the member name (for example, TCPIPX).
- If the member name on the START command is qualified by a started task identifier (for example, S TCPIPX.ABC), the jobname will be the started task identifier (for example, ABC). The started task identifier is not visible to all MVS components, but TCP/IP uses it to build the RACF resource name.
- The JOBNAME parameter can also be used on the START command to identify the jobname (for example, S TCPIPX,JOBNAME=XYZ).
- The JOBNAME can also be included on the JOB card.

In this example, user ID *USER1* is being permitted to invoke the MODDVIPA utility (and thus the SIOCSVIPA ioctl).

If this RACF profile is created, the user ID must be permitted to access this profile or else the SIOCSVIPA ioctl (and thus the MODDVIPA utility) will fail with a 'permission denied' error, regardless of SuperUser authority.

Also note that before the RACF profiles take effect, a refresh of these profiles might be required. This can be accomplished by the following RACF command:

```
SETROPTS RACLIST(SERVAUTH) REFRESH
```

For more information, refer to *z/OS Security Server RACF Security Administrator's Guide*.

Choosing which form of Dynamic VIPA support to use

The following sections explain which of the new features should be used for the type of application being used.

When should VIPADEFINE and VIPABACKUP be used to define a Dynamic VIPA?

- One or more applications bind to INADDR_ANY and exist on multiple TCP/IPs.
- Dynamic VIPA Takeover is desired.
- The DVIPA does not need to be deleted when the application is stopped.

When should VIPARANGE and bind() be used to define a Dynamic VIPA?

- The application cannot bind to INADDR_ANY or Dynamic VIPA Takeover is not desired.
- The IP address to which the application binds can be controlled by the user. The application's first explicit bind (the listening socket) will remain for the life of the application. Otherwise, the DVIPA will be removed everytime the application's DVIPA owning socket is closed, and re-added everytime there is a new DVIPA owning socket (another explicit bind has been done and the DVIPA does not exist).
- Automatic deletion of the Dynamic VIPA when the application is stopped is acceptable.
- A specific Dynamic VIPA address must be associated with a specific application.
- The application is not APF authorized, or not run under a user ID with SuperUser authority.

When should VIPARANGE and the MODDVIPA utility (or ioctl command 'SIOCSVIPA') be used to define a Dynamic VIPA?

- The application cannot bind to INADDR_ANY or Dynamic VIPA Takeover is not desired.
- The IP address to which the application binds is known but cannot be controlled by the user.
- Automatic deletion of the Dynamic VIPA when the application is stopped is not acceptable.
- The MODDVIPA utility (or application issuing the ioctl command) will be run from an APF authorized library and under a user ID with SuperUser authority.

Configuring Distributed DVIPAs — Sysplex Distributor

A Distributed DVIPA exists on several stacks, but is advertised outside the sysplex by only one stack. This stack receives all incoming connection requests and routes them to all the stacks in the distribution list for processing. This provides the benefit of distributing the workload of incoming requests and providing additional fail-safe precautions in the event of a server failure.

You can distribute connections destined for a Dynamic VIPA (DVIPA) by adding a VIPADISTribute configuration statement for a previously defined Dynamic VIPA. The order of the statements is important. The VIPA is first VIPADEFined and then VIPADISTributed. Another TCP/IP can act as a backup for the Distributed DVIPA by properly coding a VIPABackup statement; the backup will perform the routing function in the event of a failure. The options specified on a VIPADISTribute statement are inherited by a backup stack unless the second stack has its own VIPADISTribute statement, in which case it will use that VIPADISTribute statement for distributing. You can also code a VIPADISTribute statement with just the VIPABackup statement and not for the VIPADEFine statement. This would allow workload distribution only during a primary outage.

You can change the distribution of a DVIPA after a backup stack has activated it. However, if the backup stack did not not have its own distribution defined by a VIPADISTRIBUTE statement before it activated the DVIPA, any distribution changes made while the DVIPA is active on the backup stack are temporary. Those changes will be in effect while the DVIPA remains active on the backup stack, but will not be remembered if this stack takes over the DVIPA again in the future.

Following is an example of a properly coded Distributed VIPA:

```
IPCONFIG SYSPLEXROUTING DATAGRAMFWD DYNAMICXCF 193.9.200.4 255.255.255.240 1
VIPADYNAMIC
  VIPADEFINE  255.255.255.192 9.67.240.02
  VIPADISTRIBUTE DEFINE  9.67.240.02 PORT 20 21 8000 9000 DESTIP
          193.9.200.2
          193.9.200.4
          193.9.200.6
ENDVIPADYNAMIC
```

To enable the TCP/IP to forward connections, Datagram Forwarding must be enabled (specify DATAGRAMFWD in the IPCONFIG statement). There are several configuration changes that can be made to affect the method the distributing stack will use to forward connections to the target stacks. In each of the following items, *all participating stacks* is used to refer to the distributing stack and all target stacks.

WLM-based forwarding
: To enable the distributing stack to forward connections based upon the workload of each of the target stacks, configure all participating stacks for WLM GOAL mode and specify SYSPLEXROUTING in the IPCONFIG statement in all participating stacks. This will register all participating stacks with WLM and will allow the distributing stack to request workload information from WLM.

WLM/QoS-based forwarding
: To enable the distributing stack to forward connections based upon a combination of workload information and network performance information (TCP retransmissions and timeouts), configure all participating stacks for WLM GOAL mode, specify SYSPLEXROUTING in the IPCONFIG statement in all participating stacks and also define a Sysplex Distributor Performance Policy on the target stacks. For information on configuring these policies, see “Sysplex Distributor policy example” on page 573.

Random forwarding
: In the absence of any of the above configuration changes, the distributing stack will randomly select one of the target stacks for each connection.

Whether the distributing stack is performing WLM-based forwarding, WLM/QoS-based forwarding, or random forwarding, Sysplex Distributor Routing Policies can further affect the distribution of connections. Sysplex Distributor Routing Policies, configured on the distributing stack, are used to specify a set of target stacks for a given set of traffic. For example, all traffic destined to a given port/DVIPA from a specified subnet can be assigned one group of target stacks, while traffic for the same port/DVIPA from another subnet can be assigned to a different group of target stacks. For more information on configuring these types of policies, see “Sysplex Distributor policy example” on page 573.

When some targets are running WLM COMPAT mode and some are running WLM GOAL mode, the target stacks running WLM COMPAT mode will not be selected to service any requests. Only WLM GOAL mode targets will be selected when both COMPAT and GOAL modes exist.

Each distributing stack and each target stack must have a DYNAMICXCF address. When using Sysplex Distributor, do not define an IUTSAMEH link. These links will be created automatically from the DYNAMICXCF statement. This address is used by other distributing stacks as a destination point. Refer to *z/OS Communications Server: IP Configuration Reference* for directions for coding DYNAMICXCF on the IPCONFIG statement. For more information on additional configuration parameters required, also see the usage notes related to the DYNAMICXCF parameter under the IPCONFIG statement in *z/OS Communications Server: IP Configuration Reference*.

The VIPADISTribute statement specifies how new connection requests are routed to a set of candidate target stacks. The VIPADISTRIBUTEd DVIPA is followed by up to four ports, in this case the well-known ports for FTP and the ports for a custom application. Up to 32 target TCP/IPs follow the DESTIP keyword and are identified by their respective Dynamic XCF IP addresses. The VIPADISTribute statement may also specify DESTIP ALL, in which case all current and future stacks with activated Dynamic XCF may participate in the distribution as candidate target stacks. As an application listens to one of the specified ports on each listed TCP/IP, the routing TCP/IP begins to forward connections to that stack.

Sysplex wide source VIPA

Sysplex Distributor addresses the requirement of providing to clients outside a parallel sysplex a single-IP-address appearance to application instances spread across the sysplex, and also the distribution of the incoming work among the various instances. Many applications are part of a cooperative network of applications, and the sysplex applications that serve as clients to end users might also have to initiate (client-like) outbound connection requests to cooperating applications. The SOURCEVIPA feature allows applications to attain independence of any physical adapter, but SOURCEVIPA is limited to statically defined VIPAs within a stack. Different instances of the same application using Sysplex Distributor, and thus having a single IP address for inbound connection requests, will use different IP addresses for their outbound connection requests.

These problems are resolved by allowing a single sysplex wide Dynamic VIPA (DVIPA) to be used as the source IP address for TCP applications and to have the sysplex stacks collaborate on assigning ephemeral ports to prevent duplicate connection 4-tuples (combination of source and destination IP addresses and ports). These solutions are provided by sysplex wide dynamic source VIPAs for TCP connections and SYSPLEXPORTS.

Sysplex wide dynamic source VIPAs for TCP connections

The TCPSTACKSOURCEVIPA keyword on the IPCONFIG statement allows users to specify a single DVIPA to be used as a source IP address for TCP applications that initiate outbound connections on that stack. TCPSTACKSOURCEVIPA is only in effect when SOURCEVIPA is enabled and an application issues a connect() without a bind(). TCPSTACKSOURCEVIPA overrides other forms of source IP selection for all TCP applications that issue the connect() without a bind().

If you specify TCPSTACKSOURCEVIPA and do not specify SOURCEVIPA in a profile, a warning message is issued and TCPSTACKSOURCEVIPA will not be enabled. Also note that, while specifying a DVIPA as the TCPSTACKSOURCEVIPA address is most useful, any IP address in the home list can be used. Furthermore, the address specified does not need to be active on the stack at profile processing time. For example, a valid TCPSTACKSOURCEVIPA address can be an address

that falls within a VIPARANGE statement [and can be created with the MODDVIPA utility or an application bind() request], or an address that will be a target address on this stack for sysplex distribution.

Rules governing conflicts and takeover for this DVIPA activation are the same as the current rules for DVIPAs created by an application issuing bind(). This includes the function that the DVIPA will be deleted when the application closes the socket. For best performance, or to avoid the DVIPA being deleted when the first application to use it closes its socket, the system programmer might want to use the MODDVIPA utility to activate the DVIPA.

If the IP address is not an active DVIPA on the stack and cannot be made active as an application-initiated DVIPA (VIPARANGE), then the connect() call will go through normal source IP selection. A warning message will be issued no more than once every five minutes (to avoid flooding the system console), indicating an attempt to use the address specified in TCPSTACKSOURCEVIPA failed.

TCPSTACKSOURCEVIPA can be coded on all target stacks. The target TCPSTACKSOURCEVIPA statements can specify individual unique addresses, or can be duplicates of those specified on the distributing stack (a target DVIPA). Specifying the same DVIPA address for TCPSTACKSOURCEVIPA on the distributor and all target stacks creates a sysplex wide dynamic source VIPA and raises the concern for coordination of ephemeral ports across the sysplex.

For information on diagnosing sysplex wide dynamic source VIPAs for TCP connections problems, see *z/OS Communications Server: IP Diagnosis*.

SYSPLEXPORTS

Whenever two or more application instances use the same source IP address and initate connections to the same destination IP address and port, sysplex wide coordination of assignment of ephemeral ports is required so that the 4-tuple for each connection remains unique. As long as the source IP address is on a single stack, this coordination is not a problem because the stack manages assignment of ephemeral ports. However, with Sysplex Distributor applications, multiple application instances might desire to initiate connections using the same distributed DVIPA, potentially to the same destination IP address and port, so uniqueness of the connection 4-tuples cannot be guaranteed unless the stacks collaborate across the sysplex for ephemeral port assignment for distributed DVIPAs. This can be done by adding the optional SYSPLEXPORTS parameter to the VIPADISTRIBUTE statement.

SYSPLEXPORTS must be specified on the first VIPADISTRIBUTE statement processed for a particular DVIPA. It cannot be enabled once a DVIPA has been configured for distribution. Once enabled, it cannot be disabled until all distribution has been deleted for the DVIPA.

When a distributed DVIPA can be active on more than one target stack, SYSPLEXPORTS can be specified to cause the stacks to collaborate in the assignment of ephemeral ports for outbound initiated TCP connections. This ensures that two different connections do not end up with the same connection 4-tuple.

At profile processing time, a stack whose profile contains a SYSPLEXPORTS parameter on a VIPADISTRIBUTE statement will connect to the coupling facility EZBEPORT structure containing sysplex port assignment information. (The structure will be a list structure with an entry for each DVIPA address with a

VIPADISTRIBUTE with SYSPLEXPORTS specified anywhere in the sysplex. The first stack to connect to the EZBEPORT structure for a particular DVIPA will create an entry for that DVIPA in the coupling facility.) The stack will create and initialize, in the EZBEPORT structure, a sublist for this DVIPA of assigned ports for this stack.

The stack also maintains a list of allowable ephemeral ports on this stack, which is basically any port number above 1023 that has not been reserved for TCP by a PORT or PORTRANGE statement. Only port number values in this list will be allocated for use by this stack for its SYSPLEXPORTS DVIPAs. Since this list is unique to a particular stack and determined by stack configuration, a port number that is not permissible for one stack because it has been reserved might be allowable for another stack, and could in fact be allocated for use by that stack for a SYSPLEXPORTS DVIPA.

When an application issues a TCP bind() with port 0 or a connect() request, and the bind() or connect() request uses a distributed DVIPA as the source address (whether by the application explicitly binding the socket to the designated DVIPA or by the stack assigning the TCPSTACKSOURCEVIPA address) and the Distributed DVIPA is designated as SYSPLEXPORTS, TCP/IP will receive an unassigned port from the coupling facility structure that is allowable as an ephemeral port on the stack (not otherwise reserved by PORT or PORTRANGE). The stack will assign that ephemeral port as the source port for the TCP connection request, and the coupling facility structure will be updated to show the port as assigned.

This means that the maximum number of simultaneously active outbound connections using sysplex wide ephemeral port assignment is approximately 63000 for a particular distributed DVIPA, and is exactly equal to all port numbers between 1024 and 65535 that have not been reserved with a PORT or PORTRANGE configuration statement on all stacks at the same time. This is the same as for ephemeral port assignment within a single stack. That is, a single z/OS TCP stack supports no more than about 63000 simultaneously active, locally initiated TCP connections whose source ports are ephemeral ports assigned by the stack. If a stack is unable to successfully obtain an ephemeral port from the coupling facility for a SYSPLEXPORTS DVIPA, the connection request will be terminated with an error indication.

When a connection ends, and the connection's ephemeral port was a sysplex wide ephemeral port, the stack will update the coupling facility structure entry for that SYSPLEXPORTS DVIPA to indicate that the specific ephemeral port is once again available for assignment.

For information on diagnosing SYSPLEXPORTS problems, see *z/OS Communications Server: IP Diagnosis*.

Sysplex Wide Security Associations

Sysplex Wide Security Associations (SWSA) is enabled by the addition of the subparameter DVIPSEC to the FIREWALL parameter on the IPCONFIG statement. To take advantage of the functions described here, you must add this subparameter to your primary (including Sysplex Distributor hosts) and backup hosts. It is not necessary to add DVIPSEC to hosts that serve only as targets for Sysplex Distributor. For more information on configuring SWSA, see *z/OS Communications Server: IP Configuration Reference*.

SWSA also requires the use of a coupling facility structure, EZBDVIPA. For information on the setup and use of the EZBDVIPA coupling facility structure, see *z/OS Communications Server: SNA Network Implementation Guide*.

Dynamic IPSec security associations (SA), negotiated by IKE, can use a DVIPA address as the SA endpoint. Manually configured SAs are not supported by SWSA. For more information on IPSec, refer to *z/OS Security Server Firewall Technologies*.

When using SWSA, there are two possible configurations to consider:

- DVIPA takeover
- Sysplex Distributor

To support IPSec in conjunction with DVIPA takeover and Sysplex Distributor, some IKE and IPSec configuration is required. Loss of access to the coupling facility is also discussed in the following subsections.

For information on diagnosing SWSA problems, see *z/OS Communications Server: IP Diagnosis*.

DVIPA takeover

When a DVIPA is moved during DVIPA takeover (planned or unplanned), SWSA automatically re-establishes new IPSec SAs with the same security service characteristics as the SAs that existed on the host that previously owned the DVIPA. The SA re-establishment is transparent to the client that owns the other end of the SA. That is, the SA re-establishment looks like a normal SA refresh. For example, as shown in Figure 36 on page 230, during DVIPA takeover, DVIPA 192.168.253.4 is taken over by the backup host, and SAs are transparently re-established between the client and the backup host.

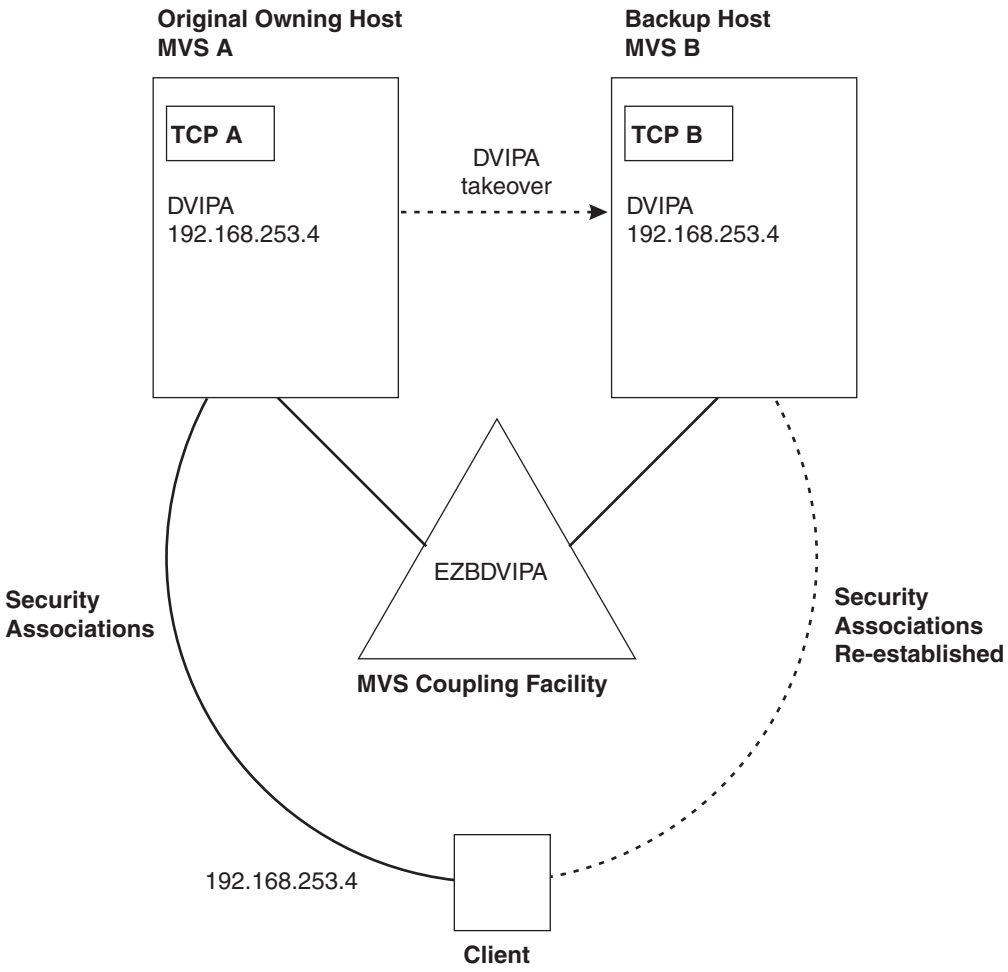


Figure 36. DVIPA takeover with SWSA

The IKE running on behalf of the TCP stack of the DVIPA owner is responsible for all IKE SA negotiations. The TCP stack owning the DVIPA is responsible for keeping the coupling facility updated with information needed to re-establish the SAs in the event of a DVIPA takeover. When a takeover occurs, the IKE on the backup host assumes responsibility for renegotiating new SAs based on the stored information read from the coupling facility during the takeover by the TCP stack of the new DVIPA owner.

Sysplex Distributor

TCP traffic protected by an IPSec SA with a sysplex-distributed DVIPA endpoint can be distributed to target hosts. IPSec cryptography for inbound traffic is performed on the target host whenever possible. If not possible, the distributor performs the cryptography before forwarding the packet to the target stack. IPSec cryptography for outbound traffic is performed on the target host, and then sent directly into the network without being routed through the distributor. Figure 37 on page 231 shows the target stack performing the cryptography for the inbound and outbound traffic.

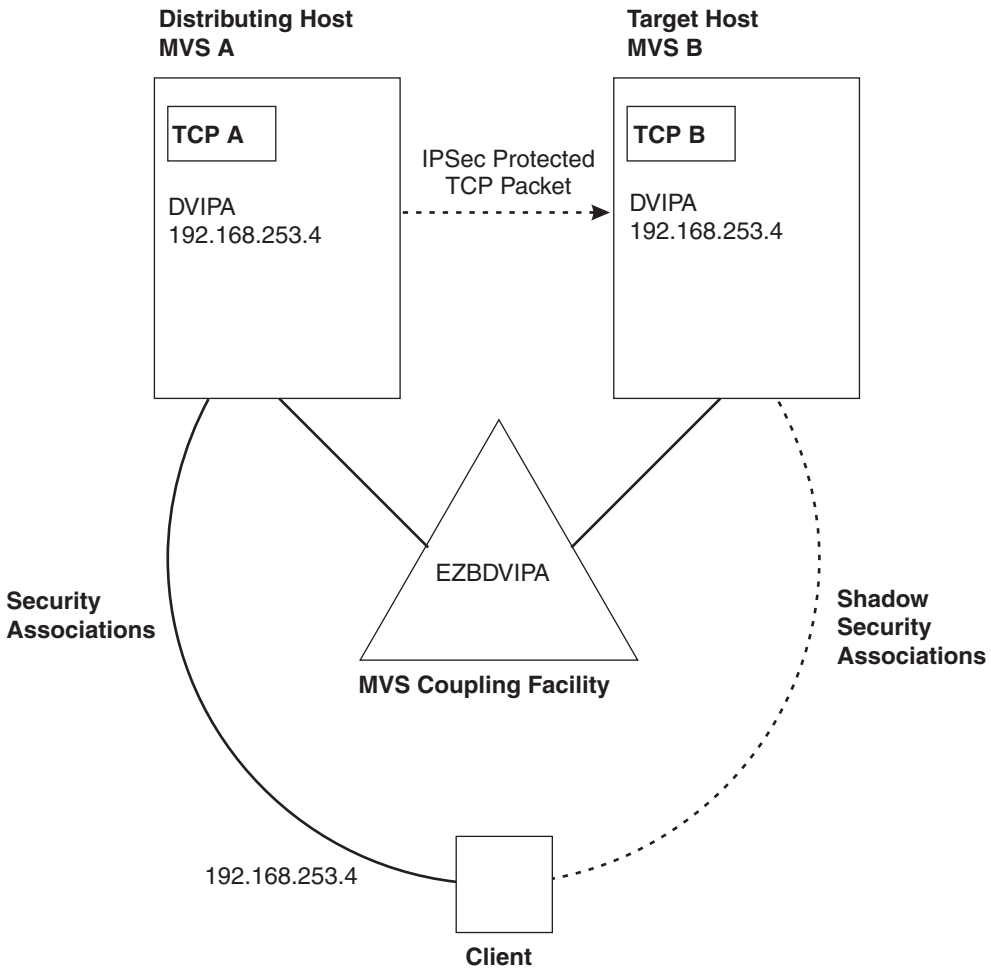


Figure 37. Sysplex Distributor with SWSA

The IKE running on behalf of the distributor TCP stack (the DVIPA owner) is responsible for all IKE SA negotiations. The distributor stack keeps the master copy of the SA associated with the DVIPA. Whenever a new SA is negotiated or refreshed and the SA is installed in the distributor stack, a copy (shadow) of the SA, which contains information necessary to perform IPSec cryptography, is sent within the sysplex to the target hosts. The shadow SAs enable the distribution of cryptography to the target stacks. The coupling facility is used as a central repository for SA replay protection sequence numbers used for outbound operations. The SA lifesizes (bytes sent and received over an SA) are maintained in the master SA.

Using IPSec with DVIPAs and Sysplex Distributor

To support IPSec in conjunction with DVIPA takeover and Sysplex Distributor, some IKE and IPSec configuration on the original or distributing host must be replicated onto all systems that can either serve as a backup host for a VIPA takeover or a target host for Sysplex Distributor. This includes IP Security policy that affects traffic using DRVIPA (from an IKE definition perspective).

- From a stack perspective, all anchor rules that are applicable to DRVIPA traffic must be identical on all systems. In addition, the ordering of the rules must allow for consistent application of security policy on all systems.
- To be considered a sysplex wide SA, the SA negotiated that applies to DVIPAs must be at a granularity no coarser than host for the local address. That is, a

dynamic SA cannot use a subnet or range that encompasses a DVIPA address. This rule ensures that on a DVIPA Giveback the SA can be moved from host to host without concerns about an SA being applicable to both the backup and primary host simultaneously. If such a dynamic SA is negotiated, the IPSec traffic using it cannot be distributed or recovered through the DVIPA takeover support.

Loss of access to coupling facility

If access is lost to the coupling facility containing the DVIPA structure EZBDVIPA, it is possible the TCP connections using this DVIPA could terminate and new connections needing IPsec will fail to establish. Loss of access could be caused by any of the following:

- A disconnect from the coupling facility structure.
- The structure is rebuilt.
- The structure encounters a critical storage shortage.

Loss of coupling facility access should only affect connections that are being encrypted or authenticated and whose filter rule is defined at a host-based granularity (no ports defined). Once access to EZBDVIPA is restored, the sessions can be re-established.

Resolution of Dynamic VIPA conflicts

The same Dynamic VIPA can exist on more than one stack in the Sysplex, playing different roles on the different stacks. The TCP/IP stacks collaborate to prevent conflicting definitions. For example, at any given time only one stack will advertise a given Dynamic VIPA to the routers.

Potentially conflicting Dynamic VIPA definitions can arise during profile processing or as the result of changes within the sysplex due to a stack or application failure or as the result of movement of workload to a different stack. The following scenarios are examples of dynamic VIPA conflict resolution handled automatically by the TCP/IP stacks. For a summary of dynamic VIPA conflict identification and resolution, see “Dynamic VIPA creation results” on page 236.

Restart of the original VIPADEFINE TCP/IP after an outage

When a dynamic VIPA is defined using VIPADEFINE on one TCP/IP, and other stacks are designated as backup using VIPABACKUP statements for the same dynamic VIPA, the stack with the highest backup rank for that DVIPA will activate it if or when the VIPADEFINE stack fails.

If the failed stack is later restarted with the same VIPADEFINE profile statement, it is likely that connections to that DVIPA will exist on the backup stack that now has the DVIPA activated and advertised to the routers. How and when ownership of the DVIPA is returned to the restarted stack is determined by how the DVIPA was originally configured.

VIPADEFINE MOVEABLE IMMEDIATE

If the DVIPA was originally configured with MOVEABLE IMMEDIATE, the following occurs:

- The DVIPA ownership is immediately transferred to the restarting stack which adds the DVIPA to its HOME list and the routers are dynamically notified. The restarted stack receives all new connections for that DVIPA. The stack also can receive packets for existing connections, and it routes these to the backup stack to preserve those connections.

- At the same time, the backup stack notifies the routers that it no longer is the owner of the DVIPA.
  - If there are no current connections to the DVIPA, it is removed from the HOME list on the backup stack and it reverts to backup status.
  - If there are any existing connections, the DVIPA remains in the HOME list of the backup stack and the DVIPA is put into *Moving* status until the last existing connection is terminated. At that time, the DVIPA is removed from the HOME list and reverts to backup status.
- IBM recommends this form of a planned DVIPA take back occur only during low periods of connection activity. This gives the attached routers time to update their routing tables and avoid connections being reset due to receiving an ICMP_HOST_UNREACH from the router.

Notes:

1. To ensure preservation of existing connections on the prior owning stack, you must define DYNAMICXCF and DATAGRAMFWD on the IPCONFIG statement.
2. MOVEABLE IMMEDIATE is the default for V2R10 and later.
3. The behavior described for MOVEABLE IMMEDIATE applies only when both the backup and the restarted stack are running V2R10 or later. If either the restarted stack or the backup stack is running V2R8, the behavior is the same as described in "VIPADEFINE MOVEABLE WHENIDLE".

VIPADEFINE MOVEABLE WHENIDLE

If the DVIPA was originally configured with MOVEABLE WHENIDLE (or the restarted or backup stack is running V2R8), the following occurs:

- If it appears that there are no active connections to the DVIPA on the backup stack:
  - The DVIPA is removed from the HOME list on the backup stack and reverts to backup status.
  - The restarted stack assumes ownership of the DVIPA by adding it to its HOME list and notifying the routers.
- If there are existing connections to the DVIPA on the backup stack:
  - Ownership of the DVIPA remains with the backup stack. The DVIPA on the restarting stack is placed in backup status at the head of the backup list for the DVIPA.
  - The backup stack periodically checks to see if it has any active connections to the DVIPA.

    When or if it appears that there are no active connections for the DVIPA, the following occurs:

    - The DVIPA is removed from the HOME list on the backup stack and reverts to backup status.
    - The restarted stack assumes ownership of the DVIPA by adding it to its HOME list and notifying the routers.

    Notes:

    1. A small period of time exists between the check for connections and the movement of the dynamic VIPA to the restarted stack. If connections are made to the old host (the backup stack) in this interval, they will be broken.
    2. During the time that TCP/IP is periodically checking for connections, TCP/IP does *not* refuse new connections because this would be the same as an outage. If moving the work back to the restarted stack is more important than maintaining uninterrupted service to all clients, then the

system operator can use VARY TCPIP,,OBEYFILE to delete the dynamic VIPA on the backup stack with the VIPADELETE profile statement. This causes the restarted stack to immediately activate the DVIPA. (Optionally, the OBEYFILE data set can contain a VIPABACKUP statement following the VIPADELETE statement. This will restore the stack as a backup stack.)

Movement of unique application-instance (BIND)

A dynamic VIPA is created when any application binds to a nonexistent, specific IP address falling within a configured VIPARANGE on that stack.

In the case of a stack failure, the same application could be started on another stack and (assuming the new stack also has an appropriate VIPARANGE configured) when the application binds to the same IP address, the dynamic VIPA is created on the second stack. Future client connections to that IP address are routed to the second stack where the application is now running.

However, if the same (or a different) application is started on a second stack and attempts to create the same dynamic VIPA using a bind() while it exists on the first stack, the end result is determined by how the VIPARANGE was configured on the stack where the first bind() occurred.

VIPARANGE (DEFINE) MOVEABLE NONDISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE NONDISRUPTIVE, the following occurs:

- The DVIPA ownership is immediately transferred to the second stack which adds the DVIPA to its HOME list and dynamically notifies the routers. This stack will now receive all new connections for the DVIPA.
- At the same time, the first stack notifies the routers that it no longer is the owner of the DVIPA, and puts the DVIPA into *moving* status. The DVIPA remains in *moving* status (and in the first stacks HOME list) until the application closes the socket.
- Existing connections on the first stack are preserved. If the second stack receives packets intended for existing connections, it routes the packets to the first stack.

Notes:

1. To ensure preservation of existing connections on the prior owning stack, you must define DYNAMICXCF and DATAGRAMFWD on the IPCONFIG statement.
2. NONDISRUPTIVE is the default for V2R10 and later.
3. The applications that create dynamic VIPAs via BIND() do *not* have to be authorized (so you might want to specify DISRUPTIVE).
4. Both stacks must be running V2R10 or later to get non-disruptive behavior. If either stack is running V2R8, the result will be as described in “VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE”.

VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE DISRUPTIVE (or if either stack is running V2R8), the following occurs:

- The bind() request for the application on the second stack will fail.
- The DVIPA on the first stack is not affected.

Note: If movement of the application from the first to the second stack is intended, the application must be ended on the first stack before it is started on the second stack.

Movement of a unique APF-authorized application instance (ioctl)

APF-authorized applications running under a user ID with SuperUser authority have the ability to activate a Dynamic VIPA with the SIOCSVIPA ioctl() either within the application itself or by invoking the MODDVIPA utility. Because this is a controlled environment, it is assumed configuration errors are minimized or avoided and the usage is correct. Thus, even if the requested DVIPA is currently active on another TCP/IP stack via BIND() or ioctll(), the DVIPA will be immediately activated on this stack. What happens on the other stack is determined by how the VIPARANGE was configured on that stack.

VIPARANGE (DEFINE) MOVEABLE NONDISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE NONDISRUPTIVE, the following occurs:

- The DVIPA ownership is immediately transferred to the second stack which adds the DVIPA to its HOME list and dynamically notifies the routers.
- At the same time, the first stack notifies the routers that it no longer is the owner of the DVIPA, and puts the DVIPA into *moving* status. The DVIPA remains in *moving* status (and in the first stack's HOME list) until the DVIPA is deleted on that stack via the VIPADELETE profile statement or the SIOCSVIPA ioctl DELETE option.
- Existing connections on the first stack are preserved. If the second stack receives packets intended for existing connections, it will route the packets to the first stack.

Notes:

1. NONDISRUPTIVE is the default for V2R10 and later.
2. Both stacks must be running V2R10 or later to get non-disruptive behavior. If either stack is running V2R8, the result will be as described in "VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE".

VIPARANGE (DEFINE) MOVEABLE DISRUPTIVE

If the first stack is configured with VIPARANGE MOVEABLE DISRUPTIVE (or if either stack is running V2R8), the following occurs:

- The ioctl request for the application on the second stack succeeds. The DVIPA is added to the HOME list on the second stack, and the routers are dynamically notified.
- The DVIPA on the first stack is deleted.

Note: Any existing connections to the DVIPA on the first stack are broken.

Same Dynamic VIPA as VIPADEFINE and BIND(), SIOCSVIPA ioctl, or MODDVIPA utility

Regardless of careful implementation, it is possible that the same IP address is inadvertently selected for VIPADEFINE and for use with BIND(), SIOCSVIPA ioctl, or the MODDVIPA utility. Because the application scenarios are quite different, this must be an error.

If this duplicate DVIPA address conflict occurs on the same TCP/IP, the second attempt might fail. If an IP address is specified in a VIPADEFINE and that same IP address has already been activated on the TCP/IP by an application via BIND(), the

SIOCSVIPA ioctl, or the MODDVIPA utility is used, the VIPADEFINE will be rejected during VARY TCPIP,,OBEYFILE processing. If an IP address is activated via VIPADEFINE, and the application does a BIND(), ioctl(), or the MODDVIPA utility is used, the BIND() will succeed, but the ioctl() will fail with a nonzero errno and the MODDVIPA utility will set a nonzero condition to indicate that the IP address already exists.

The same situation could also occur on two different TCP/IPs in the sysplex. Because the TCP/IPs are exchanging information among themselves, if the two attempts are far enough apart in time, the second attempt will be caught immediately and rejected. However, it is possible that the attempt will be made almost simultaneously on two different TCP/IPs, such that neither TCP/IP is yet aware of the attempt on the other TCP/IP. If both attempt such an activation, and the exchange of information then shows a conflict, the internal sysplex time stamps are used to determine which attempt was really first. The one that appears to be first is allowed to continue, and the Dynamic VIPA is deleted from the *later* TCP/IP. While such a simultaneous attempt is somewhat unpredictable in respect to which one will succeed, the Dynamic VIPA will remain active on only one TCP/IP, and examination of messages will indicate which TCP/IP successfully created the DVIPA and on which TCP/IP it was rejected.

Dynamic VIPA creation results

Table 12 summarizes the results of attempting to create a Dynamic VIPA when it (or the same IP address for HOME statement) already exists in the sysplex.

Table 12. Summary of Dynamic VIPA creation results

First action	Second action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
bind()	bind()	Second bind() succeeds, but no new VIPA is created.	If both stacks are running V2R10 or later, and the first BIND DVIPA was created with MOVEABLE NONDISRUPTIVE: • On stack 2, bind() succeeds • On stack 1, the BIND VIPA remains in the HOME list (unadvertised) and any existing connections are preserved • New connections to that IP address go to the application on stack 2. Otherwise, second bind fails.
bind()	ioctl()	ioctl() fails with warning condition code, but the application associated with the ioctl is still able to use the Dynamic VIPA.	ioctl() succeeds, bind is deleted (even if BIND DVIPA was created as MOVEABLE NONDISRUPTIVE)
bind()	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
bind()	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.

Table 12. Summary of Dynamic VIPA creation results (continued)

First action	Second action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
bind()	HOME	See note.	See note.
ioctl()	bind()	bind() succeeds, no new VIPA is created.	bind() fails.
ioctl()	ioctl()	Second ioctl() fails with warning condition code, but the application associated with the ioctl is still able to use the Dynamic VIPA.	Second ioctl() succeeds. If both stacks are running V2R10 or later, and the ioctl DVIPA on stack 1 was created with MOVEABLE NONDISRUPTIVE, the DVIPA on stack 1 remains in the HOME list (unadvertised) and any existing connections are preserved. Otherwise, the ioctl DVIPA on stack 1 is deleted and any existing connections are broken.
ioctl()	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
ioctl()	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
ioctl()	HOME	See note.	See note.
VIPADEFINE	bind()	bind() succeeds, but no new VIPA is created.	bind() fails.
VIPADEFINE	ioctl()	ioctl() fails.	ioctl() fails.
VIPADEFINE	VIPADEFINE	VIPADEFINE fails if the VIPADEFINE statement specifies a different mask or MOVEABLE setting than the first VIPADEFINE specified. If the second VIPADEFINE statement is an exact duplicate of the first, the second VIPADEFINE is ignored with no error message.	Second VIPADEFINE succeeds but activation on stack 2 might be deferred. If both stacks are running V2R10 or later, and the DVIPA was created on stack 1 as MOVEABLE IMMEDIATE: • Second VIPADEFINE is activated immediately • Any connections to the DVIPA on stack 1 are preserved. (DVIPA stays in HOME list unadvertised) Otherwise, the second VIPADEFINE activation is deferred until there are no connections on stack 1, at which point, stack 1 reverts to backup status.
VIPADEFINE	VIPABACKUP	VIPABACKUP fails.	Both succeed.
VIPADEFINE	HOME	See note.	See note.

Table 12. Summary of Dynamic VIPA creation results (continued)

First action	Second action	Result if second action is on the same stack	Result if the second action is on a different stack within the sysplex
VIPABACKUP	bind()	bind() fails.	If the IP address is already active on the bind() stack, the bind() will succeed. Otherwise, the bind() fails.
VIPABACKUP	ioctl()	ioctl() fails.	ioctl() fails.
VIPABACKUP is backup status	VIPADEFINE	VIPADEFINE succeeds, replaces the VIPABACKUP.	VIPADEFINE succeeds.
VIPABACKUP in active status (after takeover)	VIPADEFINE	VIPADEFINE rejected	**Note:** The VIPABACKUP DVIPA is MOVEABLE IMMEDIATE or WHENIDLE depending how the original VIPADEFINE DVIPA was created. If both stacks are running V2R10 or later, and the VIPABACKUP DVIPA is MOVEABLE IMMEDIATE: • The VIPADEFINE is activated immediately. • Any connections to the DVIPA on stack 1 are preserved (DVIPA stays in HOME list unadvertised). • When there are no more connections, stack 1 reverts to backup status. Otherwise, the VIPADEFINE activation on stack 2 is deferred until there is no stack 1, at which point, stack 1 reverts to backup status.
VIPABACKUP	VIPABACKUP	Second VIPABACKUP succeeds.	Second VIPABACKUP succeeds.
VIPABACKUP	HOME	See note.	See note.
HOME	bind()	bind() succeeds, but no new VIPA is created.	bind() fails.
HOME	ioctl()	ioctl() fails.	ioctl() fails.
HOME	VIPADEFINE	VIPADEFINE fails.	VIPADEFINE fails.
HOME	VIPABACKUP	VIPABACKUP fails.	VIPABACKUP fails.
Note: Defining the same IP address in the HOME statement as an existing Dynamic VIPA will not be rejected by the TCP/IP stack, but it is likely to cause routing problems.			

Other considerations

The following sections describe other considerations you should understand regarding Dynamic VIPA support.

Mixture of types of Dynamic VIPAs within subnets

Any particular IP address can be used in only one way as a Dynamic VIPA. As described in previous sections, a Dynamic VIPA can be defined either via VIPADEFINE or by application action within a valid VIPARANGE, but not both. However, within a subnet defined as a VIPARANGE, some IP addresses can be used for VIPADEFINE, and others may be assigned to unique application instances, without conflict, as long as the limit of a total of 256 active and backup Dynamic VIPAs on a single TCP/IP is not exceeded. TCP/IP will make no attempt to reject a VIPADEFINE Dynamic VIPA that also falls within a VIPARANGE. This allows installations with limited availability of IP addresses to assign individual addresses to either application scenario, without having to define separate subnets and use up additional IP addresses in that manner.

MVS failure and Sysplex Failure Management

The TCP/IPs in a Sysplex use MVS XCF Messaging to exchange information about Dynamic VIPAs. When a TCP/IP fails or is ended by operator command, but the underlying MVS remains active, the other TCP/IPs are immediately notified, and takeover of VIPADEFINE Dynamic VIPAs is automated and very fast.

However, when an MVS fails, there is normally an operator message on the console requiring a response (WTOR). Until this response is made by an operator or automation, the other MVSs do not notify the remaining TCP/IPs in the sysplex of the failure of the TCP/IP on the failing MVS. This can delay automated backup of VIPADEFINED Dynamic VIPAs. Sysplex Failure Management (SFM) can be used to automate the required response to the console message of the failing MVS. Refer to *z/OS MVS Setting Up a Sysplex* for information on how to set up SFM to avoid the requirement for a manual response and speed backup of VIPADEFINED Dynamic VIPAs.

For more information, refer to *z/OS Communications Server: IP Diagnosis*.

Applications and Dynamic VIPAs

While most applications support multiple instances in a sysplex, very few applications expect IP addresses to move around under them. TCP applications use TCP connections to form a relationship between particular client and server instances to exchange data over an extended period. They rely on notification of TCP connection termination to initiate recovery and to reestablish a new relationship (possibly from a client to a different server). Conversely, most UDP applications do the equivalent function at the application layer. Movement of an IP address to a different server could be confusing to the client, unless the new server also is aware of the state of the client work.

UDP applications whose interactions consist of atomic interactions (a single request followed by one or more responses, with no state information maintained at the server between requests) can use Dynamic VIPAs in the Multiple Application-Instance Scenario. However, if the server application maintains state information between interactions (for example, NFS), then moving a Dynamic VIPA to another server might not work unless the client/server application protocol can

detect the discontinuity. In that case, the Unique Application-Instance Scenario might apply, which would require the restart of the server instance on another TCP/IP.

In addition, the following types of work are not appropriate for distribution with distributed dynamic VIPA:

- Applications that establish affinity with a particular TCP/IP stack, such as SNMP.
- Applications that bind to ephemeral ports.
- FTP servers that receive the PASV command for a distributed DVIPA that did not specify SYSPLEXPORTS. The PASV command is supported when SYSPLEXPORTS was specified on the VIPADISTribute statement of the distributed DVIPA that is the destination IP address being used by the FTP server. This command requests the FTP server to bind() on a data port that is not the default data port, or the one specified on the VIPADISTribute statement, and to wait for a connection rather than initiate one on receipt of a transfer command (for example, RETR).

Example of configuring Dynamic and Distributed VIPAs

The TCP/IP profiles needed to implement Dynamic VIPA(DVIPA) on multiple systems in a sysplex are shown in the following examples. The VIPADEFINE and VIPABACKUP statements allow Automatic Dynamic VIPA Takeover to occur if needed (see “Configuring the Multiple Application-Instance Scenario” on page 219), and the VIPARANGE statements allow Dynamic VIPAs to be dynamically created by an application or by the MODDVIPA utility (see “Configuring the Unique Application-Instance Scenario” on page 219). The VIPADISTRIBUTE statements allow a single VIPA to be shared among several TCP/IPs. Including the SOURCEVIPA and TCPSTACKSOURCEVIPA parameters on the IPCONFIG statement, on each target stack with the same Dynamic VIPA specified, enables a single DVIPA address to be used as a sysplex-wide source DVIPA address for outbound TCP connections.

```
TCPCS
 IPCONFIG  DATAGRAMFWD SYSPLEXROUTING SOURCEVIPA TCPSTACKSOURCEVIPA 201.2.10.11
 DYNAMICXCF 193.9.200.1 255.255.255.240 14
 VIPADYNAMIC
  VIPADEFINE 255.255.255.240  201.2.10.11 201.2.10.12
  VIPADISTRIBUTE 201.2.10.11 SYSPLEXPORTS PORT 20 21 DESTIP ALL
  VIPADISTRIBUTE 201.2.10.12 PORT 20 21 DESTIP 193.9.200.2
  VIPABACKUP 100 201.2.10.13
  VIPABACKUP 80  201.2.10.21
  VIPABACKUP 80  201.2.10.22
  VIPARANGE DEFINE 255.255.255.192  201.2.10.192
 ENDVIPADYNAMIC
TCPCS2
 IPCONFIG  DATAGRAMFWD SYSPLEXROUTING SOURCEVIPA TCPSTACKSOURCEVIPA 201.2.10.11
 DYNAMICXCF 193.9.200.2 255.255.255.240 1
 VIPADYNAMIC
  VIPADEFINE 255.255.255.192  201.2.10.13
  VIPABACKUP 100 201.2.10.11  201.2.10.21
  VIPABACKUP 75  201.2.10.12  201.2.10.22
  VIPARANGE DEFINE 255.255.255.192  201.2.10.192
 ENDVIPADYNAMIC
TCPCS3
 IPCONFIG  DATAGRAMFWD SYSPLEXROUTING SOURCEVIPA TCPSTACKSOURCEVIPA 201.2.10.11
 DYNAMICXCF 193.9.200.3 255.255.255.240 1
 VIPADYNAMIC
  VIPADEFINE 255.255.255.192   201.2.10.21  201.2.10.22
  VIPABACKUP  10  201.2.10.11  201.2.10.12  201.2.10.13
  VIPARANGE DEFINE 255.255.255.192          201.2.10.192
 ENDVIPADYNAMIC
```

```
TCPCS6
IPCONFIG DATAGRAMFWD SYSPLEXROUTING SOURCEVIPA TCPSTACKSOURCEVIPA 201.2.10.11

DYNAMICXCF 193.9.200.6 255.255.255.224 1

TCPCS6 does not have dynamic VIPAs defined so it does not
contain a VIPADYNAMIC definition.  It has a DYNAMICXCF
statement to enable XCF dynamic support, and SOURCEVIPA and
TCPSTACKSOURCEVIPA to enable the distributed DVIPA 201.2.10.11.
```

Start TCP/IP on each system as shown above.

- On system1, start TCPCS and TCPCS2.
- On system2, start TCPCS3, on system3 start TCPCS6.
- On system1, run the MODDVIPA utility to define the DVIPA 201.2.10.193.

```
//TCPDVP   PROC
//*
//*
//TCPDVP  EXEC PGM=MODDVIPA ,REGION=0K,TIME=1440,                       x
//             PARM='POSIX(ON) ALL31(ON)/-p TCPCS -c 201.2.10.193'
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSERR   DD SYSOUT=*
//SYSERROR DD SYSOUT=*
//SYSDEBUG DD SYSOUT=*
//SYSUDUMP DD SYSOUT=A
//SYSABEND DD SYSOUT=*
//*
//*Run program here
//*
//TCPDVP  EXEC PGM=MODDVIPA ,REGION=0K,TIME=1440,                       x
//             PARM='POSIX(ON) ALL31(ON)/-p TCPCS -d 201.2.10.193'
```

The PARM field can be -c for create or -d for delete. The example above will create DVIPA 201.2.10.193 for the TCP/IP named TCPCS. After intermediate program has completed (and the comment character is removed), the DVIPA will be deleted.

Verifying the DVIPAs in a sysplex

A display command parameter displays Dynamic VIPAs in the sysplex (see Figure 35 on page 217). In the following example taken from stack TCPCS, the ORIGIN lines show that 201.2.10.11 and 201.2.10.12 were created by VIPADEFINE on this stack, 201.2.10.193 was created by VIPARANGE ioctl (issued through the MODDVIPA utility), and all of the others were created by VIPABACKUP statements. The command is:

```
d tcpip,tcpname,sysplex,vipadyn
```

The ORIGIN line indicates how the DVIPA is configured on the stack specified by `tcpname`. Each stack (TCPNAME) for each system (MVSNAME) is shown with its status (STATUS). Two other status values not shown in the following example are:

QUIESCING
: This DVIPA was a target for distribution and has been removed as a target. However, it is still servicing one or more connections for this DVIPA. The DVIPA will be removed when all connections complete.

MOVING
: This DVIPA was active on this stack and has been moved to another stack. Connections on this stack for this DVIPA prior to the move will still be serviced by this stack until completion.

The rank (RANK) indicates which of the stacks is eligible to take over if the stack on which the DVIPA is active stops. The stack with the highest rank is the one that will take over the DVIPA.

```
D TCPIP,TCPCS,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V1R4 874
VIPA DYNAMIC DISPLAY FROM TCPCS    AT MVS005
IPADDR: 201.2.10.11  LINKNAME: VIPLC9020A0B
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      BOTH
  TCPCS2   MVS004   BACKUP 100                                  DEST
  TCPCS3   MVS005   BACKUP 010                                  DEST
  TCPCS6   MVS006   ACTIVE                                      DEST
IPADDR: 201.2.10.12  LINKNAME: VIPLC9020A0C
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      DIST
  TCPCS2   MVS004   BACKUP 075                                  DEST
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.13
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS2   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 100
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.21
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS2   MVS004   BACKUP 100
  TCPCS    MVS004   BACKUP 080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 080
  TCPCS2   MVS004   BACKUP 075
IPADDR: 201.2.10.193  LINKNAME: VIPLC9020AC1
  ORIGIN: VIPARANGE ioctl
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.192 201.2.10.192
```

TCPCS2, TCPCS3, and TCPCS6 all display the same information about all the DVIPAs. ORIGIN fields are displayed for the DVIPAs that are configured on this stack.

```
D TCPIP,TCPCS2,SYS,VIPAD
EZZ8260I SYSPLEX CS V1R4 877
VIPA DYNAMIC DISPLAY FROM TCPCS2   AT MVS005
IPADDR: 201.2.10.11
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      BOTH
  TCPCS2   MVS004   BACKUP 100                                  DEST
  TCPCS3   MVS005   BACKUP 010                                  DEST
  TCPCS6   MVS006   ACTIVE                                      DEST
IPADDR: 201.2.10.12
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
```

```
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      DIST
  TCPCS2   MVS004   BACKUP 075                                   DEST
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.13  LINKNAME: VIPLC9020A0D
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS2   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 100
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.21
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS2   MVS004   BACKUP 100
  TCPCS    MVS004   BACKUP 080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 080
  TCPCS2   MVS004   BACKUP 075
IPADDR: 201.2.10.193
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.192 201.2.10.192
```

In the following example, TCPCS6 knows about the DVIPAs on the other stacks. There are no DVIPAs configured on TCPCS6, thus, no ORIGIN fields displayed.

```
D TCPIP,TCPCS6,SYS,VIPAD
EZZ8260I SYSPLEX CS V1R4 880
VIPA DYNAMIC DISPLAY FROM TCPCS6   AT MVS005
IPADDR: 201.2.10.11
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      BOTH
  TCPCS2   MVS004   BACKUP 100                                  DEST
  TCPCS3   MVS005   BACKUP 010                                  DEST
  TCPCS6   MVS006   ACTIVE                                      DEST
IPADDR: 201.2.10.12
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      DIST
  TCPCS2   MVS004   BACKUP 075                                  DEST
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.13
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS2   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 100
  TCPCS3   MVS005   BACKUP 010
IPADDR: 201.2.10.21
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS2   MVS004   BACKUP 100
  TCPCS    MVS004   BACKUP 080
IPADDR: 201.2.10.22
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS005   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 080
  TCPCS2   MVS004   BACKUP 075
```

```
IPADDR: 201.2.10.193
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.192 201.2.10.192
```

Using NETSTAT support to verify Dynamic VIPA configuration

The netstat commands (TSO NETSTAT, z/OS UNIX onetstat, and MVS console D TCPIP,,Netstat) have a VIPADCFG (-F) report and a VIPADYN (-v) report. These reports show the Dynamic VIPA configuration for a particular TCP/IP.

Note: Use the CONFIG (-f) report to verify the rest of the stack's configuration, including SOURCEVIPA and TCPSTACKSOURCEVIPA.

The Dynamic VIPA Information section is only displayed when there are DVIPAs configured on this stack. The VIPA Range section, displayed only if a VIPARANGE statement was processed in this stacks profile (or OBEYFILE data set), indicates only that a range was configured. It does not indicate whether any ioctl or BIND has actually created a DVIPA in the specified range. The VIPA Distribute section is displayed only if there are VIPADISTRIBUTE statements configured on this stack.

On stack TCPCS, using netstat on OMVS:

```
# netstat -p tcpcs -F
MVS TCP/IP onetstat CS V1R4       TCPIP Name: TCPCS            12:04:15
Dynamic VIPA Information:

VIPA Backup:
  IP Address       Rank
  ----------       ----
  201.2.10.13      000100
  201.2.10.21      000080
  201.2.10.22      000080

VIPA Define:
  IP Address       AddressMask      Moveable    SrvMgr
  ----------       -----------      --------    ------
  201.2.10.11      255.255.255.240  Immediate    Yes
  201.2.10.12      255.255.255.240  Immediate    No

VIPA Range:
  AddressMask      IP Address       Moveable
  -----------      ----------       --------
  255.255.255.192  201.2.10.192     NonDisr

VIPA Distribute:
  IP Address       Port   XCF Address   SysPt
  ----------       ----   -----------   -----
  201.2.10.11      00020  ALL            Yes
  201.2.10.11      00021  ALL            No
  201.2.10.12      00020  193.9.200.2    No
  201.2.10.12      00021  193.9.200.2    No

VIPA Service Manager:
  McastGroup: 245.10.131.201     Port: 01472  Pwd: Yes
```

On stack, TCPCS2 from the console:

```
01.55.13  d tcpip,tcpcs2,net,vipadcfg
01.55.14  EZZ2500I NETSTAT CS V1R4 TCPCS2 764
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IP ADDRESS       RANK
    ----------       ----
    201.2.10.11      000100
```

```
    201.2.10.12      000075
    201.2.10.21      000100
    201.2.10.22      000075
  VIPA DEFINE:
    IP ADDRESS       ADDRESSMASK      MOVEABLE    SrvMgr
    ----------       -----------      --------    ------
    201.2.10.13      255.255.255.192  IMMEDIATE    No
  VIPA RANGE:
    ADDRESSMASK      IP ADDRESS       MOVEABLE
    -----------      ----------       --------
    255.255.255.192  201.2.10.192     NONDISR
```

On stack TCPCS3 from the console:

```
01.56.42  d tcpip,tcpcs3,net,vipadcfg
01.56.43  EZZ2500I NETSTAT CS V1R4 TCPCS3 767
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IP ADDRESS       RANK
    ----------       ----
    201.2.10.11      000010
    201.2.10.12      000010
    201.2.10.13      000010
  VIPA DEFINE:
    IP ADDRESS       ADDRESSMASK      MOVEABLE    SrvMgr
    ----------       -----------      --------    ------
    201.2.10.21      255.255.255.192  IMMEDIATE    No
    201.2.10.22      255.255.255.192  IMMEDIATE    No
  VIPA RANGE:
    ADDRESSMASK      IP ADDRESS       MOVEABLE
    -----------      ----------       --------
    255.255.255.192  201.2.10.192     NONDISR
```

On stack TCPCS6 from the console:

```
01.57.32  d tcpip,tcpcs6,net,vipadcfg
01.57.32  EZZ2500I NETSTAT CS V1R4 TCPCS6 770
```

The VIPADYN (-v) report displays all the Dynamic VIPAs available to this stack, as shown in the following examples.

On stack TCPCS using netstat on OMVS:

```
 # netstat -p tcpcs -v
MVS TCP/IP onetstat CS V1R4       TCPIP Name: TCPCS           02:03:07
IP Address      AddressMask     Status    Origination     DistStat
----------      -----------     ------    -----------     --------
201.2.10.11     255.255.255.240 Active    VIPADefine      Dist/Dest
201.2.10.12     255.255.255.240 Active    VIPADefine      Dist
201.2.10.13     255.255.255.192 Backup    VIPABackup
201.2.10.21     255.255.255.192 Backup    VIPABackup
201.2.10.22     255.255.255.192 Backup    VIPABackup
```

On stack TCPCS2 from the console:

```
02.04.09  d tcpip,tcpcs2,net,vipadyn
02.04.09  EZZ2500I NETSTAT CS V1R4  TCPCS2 795
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 BACKUP    VIPABACKUP      DEST
201.2.10.12     255.255.255.240 BACKUP    VIPABACKUP      DEST
201.2.10.13     255.255.255.192 ACTIVE    VIPADEFINE
```

On stack TCPCS, using the onetstat command:

```
# netstat -p TCPCS2 -v
MVS TCP/IP onetstat CS V1R4        TCPIP Name: TCPCS2          10:20:58
IP Address      AddressMask     Status    Origination     DistStat
----------      -----------     ------    -----------     --------
```

```
201.2.10.11     255.255.255.240 Backup    VIPABackup      Dest
201.2.10.12     255.255.255.240 Backup    VIPABackup      Dest
201.2.10.13     255.255.255.192 Active    VIPADefine
201.2.10.21     255.255.255.192 Backup    VIPABackup
201.2.10.22     255.255.255.192 Backup    VIPABackup
```

On stack TCPCS3 from the console:

```
02.05.21  d tcpip,tcpcs3,net,vipadyn
02.05.21  EZZ2500I NETSTAT CS V1R4  TCPCS3 798
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 BACKUP    VIPABACKUP      DEST
201.2.10.12     255.255.255.240 BACKUP    VIPABACKUP
201.2.10.13     255.255.255.192 BACKUP    VIPABACKUP
201.2.10.21     255.255.255.192 ACTIVE    VIPADEFINE
201.2.10.22     255.255.255.192 ACTIVE    VIPADEFINE
```

On stack TCPCS6 from the console:

```
02.05.58  d tcpip,tcpcs6,net,vipadyn
02.05.58  EZZ2500I NETSTAT CS V1R4 TCPCS6 801
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 ACTIVE                    DEST
```

Verifying Sysplex Distributor workload

The netstat commands (TSO NETSTAT, z/OS UNIX onetstat, and MVS console D TCPIP,,Netstat) have a VDPT (-O) report and a VCRT (-V) report. Refer to *z/OS Communications Server: IP System Administrator's Commands* for more information on these commands.

Run onetstat -O on the distributing stack to confirm that there are target stacks available with server applications ready. This display will only show target stacks that are currently up and have joined the sysplex. The READY field indicates how many, if any, applications the target TCP/IP, identified by its DestXCF Addr, has bound to DPort. If none, then this target TCP/IP will not receive any connection workload. The TotalConn field indicates how many connections this distributing TCP/IP has forwarded to the target TCP/IP.

Note: TotalConn is a historical count and will wrap.

The following netstat display command on the distributing stack, TCPCS, shows which target stacks are available with the server applications ready. The target stack is identified by its Dynamic XCF address (DESTXCF ADDR). The READY field indicates how many applications on that target stack have bound to the DPORT. TOTALCONN is the number of all connections the distributing stack, TCPCS, has routed to the target stack. WLM is the Workload Manager weight value for the target TCP/IP stack.

```
d tcpip,tcpcs,net,vdpt

EZZ2500I NETSTAT CS V1R4 TCPSVT
DYNAMIC VIPA DISTRIBUTION PORT TABLE:
DEST IPADDR     DPORT DESTXCF ADDR    RDY TOTALCONN  WLM
201.2.10.11     00020 193.9.200.1     000 0000003561 01
201.2.10.11     00020 193.9.200.2     000 0000003500 01
201.2.10.11     00020 193.9.200.3     000 0000003700 02
201.2.10.11     00021 193.9.200.1     000 0000000499 01
201.2.10.11     00021 193.9.200.2     000 0000000450 01
201.2.10.11     00021 193.9.200.3     000 0000000415 02
201.2.10.12     00020 193.9.200.2     000 0000000239 01
201.2.10.12     00021 193.9.200.2     000 0000000059 01
```

The following netstat display command on the distributing stack displays all current connections being distributed by TCPCS.

```
d tcpip,tcpcs,net,vcrt

EZZ2500I NETSTAT CS V1R4 TCPCS 363
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST IPADDR      DPORT  SRC IPADDR     SPORT       DESTXCF ADDR
-----------      -----  ----------     -----       ------------
201.2.10.11      00021  193.9.200.5    01029       193.9.200.1
201.2.10.11      00021  193.9.200.8    01050       193.9.200.2
201.2.10.11      00021  193.9.200.11   01079       193.9.200.3
201.2.10.12      00021  193.9.200.9    01030       193.9.200.2
```

Dynamic VIPAs and routing protocols

With Dynamic VIPAs, IP addresses may move from one stack to another. These changes need to be communicated to the network. Therefore, dynamic routing should be implemented when Dynamic VIPAs are being used.

OMPROUTE

The names of Dynamic VIPA interfaces are assigned dynamically by the stack when a Dynamic VIPA interface is created. Therefore, the Name field set on the Interface or OSPF_Interface statement for a Dynamic VIPA will be ignored by OMPROUTE.

It is recommended that a host have an Interface or OSPF_Interface definition for every Dynamic VIPA address which that host might own. Because this could be a large number of interfaces, additional wildcard capabilities have been added to OMPROUTE, for Dynamic VIPA interfaces only.

Ranges of Dynamic VIPA interfaces can be defined using the subnet mask parameter on the OSPF_Interface or Interface statement. The range defined will be all the IP addresses that fall within the subnet defined by the mask and the IP address. The following example defines a range of Dynamic VIPA addresses from 10.138.165.80 to 10.138.165.95:

```
OSPF_Interface
   IP_address = 10.138.165.80
   Name = dummy_name (see note)
   Subnet_mask = 255.255.255.240;
```

Note: The *Name* parameter is required and must be unique, but it is not actually used for Dynamic VIPAs.

For consistency with the VIPARANGE statement in the TCP/IP profile, any value that may fall within the range can be used with the mask to define a range of Dynamic VIPAs. The interface statement in the following example has the same meaning as the one in the example above:

```
OSPF_Interface
   IP_address = 10.138.165.87
   Name = dummy_name
   Subnet_mask = 255.255.255.240;
```

Notes:

1. When defining ranges, it is not necessary or desirable to code a destination address. OMPROUTE will automatically set the destination address of a Dynamic VIPA to its IP address.
2. There is nothing in the interface definition statements that informs OMPROUTE that a particular interface definition statement is for a Dynamic VIPA or a range

of Dynamic VIPAs. Rather, OMPROUTE learns this information from the stack when these interfaces are created or taken over.

The MTU size defined on OSPF_INTERFACE statements limits the size of advertisements that can be sent or received over OMPROUTE interfaces. OMPROUTE cannot build an advertisement whose size would exceed the largest MTU size of all its interfaces. Also, OMPROUTE cannot receive an advertisement that is larger than the largest MTU size defined for all its interfaces. In either of these cases, you will see the following message:

```
EZZ7967I ADVERTISEMENT DISCARDED, OVERFLOWS BUFFER: LS
   TYPE x ID x.x.x.x ORG y.y.y.y
```

When this happens on an originating host, that host will not be able to send router Link State Advertisements (LSAs), and therefore other hosts will not be able to calculate routes to any destinations (for example, VIPAs) owned by the originating host. OMPROUTE will terminate if it encounters this condition. If it cannot send its router LSA, it is useless as a router. When this happens on a receiving host, that host will not be able to compute routes to any destinations advertised in the discarded LSA. Also note that other OSPF implementations might have similar or stricter limitations, in which case they would be unable to receive or propagate large router LSAs received from OMPROUTE. These scenarios can severely affect network connectivity and routing capability. If large numbers of VIPA interfaces are going to be used, we recommend you examine OSPF MTU sizes throughout your network to ensure that large router LSAs can be propagated.

Normally, large router LSAs would not be a problem, as LSAs seldom exceed their allowed MTU sizes. However, if a large number of VIPA or dynamic VIPA interfaces are defined on a host, this can become a consideration. The size of the router LSA will include 52 bytes for headers, plus the number of bytes required to advertise the host's owned interfaces. The number of bytes required for each interface is:

VIPA	12 bytes, plus 12 bytes for each VIPA subnet (see the following example)
Point-to-point	24 bytes
Point-to-multipoint	12 bytes, plus 12 bytes for each neighbor on the interface
All other types	12 bytes

For owned VIPA interfaces, OMPROUTE normally advertises both host and subnet routes. The size of router LSAs required can be minimized by careful subnet planning. For example, assume the following definition exists in the OMPROUTE configuration file:

```
OSPF_Interface
   IP_Address=3.3.3.*
   Name = VIPA1A
   Subnet_Mask=255.255.255.252
   Attaches_To_Area=1.1.1.1
   MTU=1024
   Cost0 = 1;
```

If 101 VIPA interfaces, numbered 3.3.3.1 to 3.3.3.101, are activated, in addition to the headers and any other owned interfaces, OMPROUTE would need 1512 bytes to advertise 126 links in its router LSA (1 host route to each of the VIPAs, plus 25 subnet routes since each subnet contains only four addresses).

By contrast, assume the following definition exists in the OMPROUTE configuration file:

```
OSPF_Interface
   IP_Address=3.3.3.*
   Name = VIPA1A
   Subnet_Mask=255.255.255.0
   Attaches_To_Area=1.1.1.1
   MTU=1024
   Cost0 = 1;
```

If the same 101 VIPA interfaces are activated, OMPROUTE would advertise 102 links in its router LSA (1 host route to each VIPA, plus 1 subnet route since all the VIPAs are in the same subnet). This would only require 1224 bytes to advertise the VIPAs. If the MTU size on the network is 1500, this can make the difference between being able to send or receive a router LSA or not being able to send or receive a router LSA. This limitiation can further be circumvented by suppressing VIPA host routes by coding SUBNET=YES on the OSPF_INTERFACE statement for the VIPA interfaces. However, there are limits on when this can be done. For details, see the *z/OS Communications Server: IP Configuration Reference*.

RIP (Routing Information Protocol)

If using RIP services and Host Route advertising is not supported by adjacent routers (that is, inability to learn host routes), the following restrictions for VIPA addresses must be applied to benefit from fault tolerance support:

- If you use subnetting and VIPA addresses are in the same network as the physical IP addresses, the subnetwork portion of any VIPA addresses must not be the subnetwork portion of any physical IP addresses in the network. In this case, assign a new subnetwork for the VIPA address.
- If subnetting is not used on any physical interface, the network portion of any VIPA addresses must not be the network portion of any physical IP addresses in the network. In this case, assign a new network for the VIPA address, preferably a class C network address.

If using RIP services and Host Route advertising is supported by adjacent routers, the network or subnetwork portions of VIPA addresses can be the same across multiple z/OS TCP/IP stacks in the network. To enable Host Route advertising in OMPROUTE, configure RIP_Interface Send_Host_Routes=YES.

Chapter 6. TCP/IP in a sysplex

The increasing demands of network servers, and in particular z/Series servers, has led to the creation of different techniques to address performance requirements when a single server is not capable of providing the availability and scalability demands placed on it by its clients. Specifically, network solutions make use of what is referred to as the clustering technique, whereby multiple servers are associated together into a cluster to provide sufficient processing power and availability characteristics to handle the demands of the clients.

In the scope of this chapter, this cluster functionality is provided by the sysplex. That is, the sysplex provides the necessary capability to cluster together a number of z/Series servers that cooperate with one another to deliver the processing power needed to service the demands required of a particular service environment.

Solutions utilizing the clustering approach to increase server availability and processing capability attempt to provide mechanisms by which they ensure the viability of the cluster in an environment containing a large number of clients generating a potentially high number of requests. To do so, the cluster technique can provide for two main objectives, high availability and load balancing. In some cases, clustering techniques address only high availability, as is the case with Dynamic VIPA that provides for availability in spite of potential TCP/IP stack or z/OS image failures. In other cases, the intent is to provide for both high availability and load balancing, as is done by the Domain Name System/Workload Manager solution (DNS/WLM) and Sysplex Distributor.

In general, load balancing refers to the ability to utilize different systems within the cluster simultaneously, thereby taking advantage of the additional computational function of each. Further, clustering techniques addressing load balancing lead to other system requirements, such as that of a single systemwide image (one identity by which clients access the system), horizontal growth, and ease of management.

The traditional view of a single server has been primarily a single machine with perhaps a few network interfaces (IP addresses). This tends to lead to many potential points of failure within the server: the machine itself (hardware), the operating system (including TCP/IP stack) kernel executing on the machine, or a network interface (and the IP address associated with it). Static Virtual IP Addresses (VIPAs) exclude the network interface as a point of failure while Dynamic VIPAs additionally aid with server (image) or kernel failure. In this way, high availability is seen as the availability of the entire server cluster and the service it provides. Further, VIPAs can be used in conjunction with the load balancing solutions discussed in this document, DNS/WLM and Sysplex Distributor.

Clustering techniques that address the load balancing of connections requests also typically provide for some high availability. That is, these techniques dispatch connections to target servers and can exclude failed servers from the list of target servers that can receive connections. In this way, the dispatching function avoids routing connections and requests to a server incapable of satisfying such requests.

Load balancing is the ability for a cluster to spread workload evenly (or based on some policy) to target servers comprising the cluster. Usually, this load balancing is measured by some notion of perceived load on each of the target servers. This chapter describes two techniques that provide load balancing: DNS/WLM and Sysplex Distributor. Each identifies the target zSeries servers willing to receive client connections based on some specification.

By providing load balancing, clustering techniques must also provide for other system requirements in addition to the dispatching of connections. These include the ability to advertise some single systemwide image or identity so that clients can uniquely and easily identify the service. Additionally, clustering techniques should also provide for horizontal growth of the system and ease of management.

There is an excellent description of sysplexes in *z/OS Parallel Sysplex Overview*. Refer to the Redbook, *TCP/IP in a Sysplex*, for more detailed information on implementing load balancing and availability in your sysplex.

Connectivity in a sysplex

With Dynamic VIPAs, IP addresses may move from one stack to another. These changes need to be communicated to the network. Therefore, dynamic routing should be implemented when dynamic VIPAs are being used. Refer to "Dynamic VIPAs and routing protocols" on page 247 for more detailed information.

Dynamic XCF

The IPCONFIG DYNAMICXCF (Dynamic XCF) statement can be used to create trusted, internal links to other stacks within a sysplex. Dynamic XCF creates a single IP address by which all other stacks in the sysplex may reach the stack. Most point-to-point links have their own unique IP address. A unique pair of IP addresses is needed for each stack within a sysplex using normal point-to-point links. This tends to use more IP addresses. IP addresses can be saved by using Dynamic XCF. Additionally, the Dynamic XCF statement automatically generates the appropriate DEVICE, LINK, HOME, BSDROUTINGPARMS and BEGINROUTES definitions (as described below) and activates the devices to enable the stack to communicate with other stacks in the sysplex.

Dynamic XCF devices and links, when activated, appear to the stack as though they had been defined in the TCP/IP profile. They can be displayed using standard commands. Dynamic XCF is activated via the DYNAMICXCF keyword on the IPCONFIG statement, which is described in detail below.

Dynamic XCF can be used to generate dynamic definitions for TCP/IP stacks that reside on another z/OS host in a sysplex and for additional TCP/IP stacks that reside on the same z/OS host.

The minimum requirements in order for TCP/IP stacks to utilize XCF Dynamics differ based on whether same host or inter-host communication is being used. In order to generate definitions for two TCP/IP stacks that reside on different MVS hosts:

- Both MVS hosts must belong to the same sysplex.
- VTAM must have XCF communications enabled by specifying XCFINIT=YES as a startup parameter or by activating the VTAM major node, ISTLSXCF. For details about configuration, refer to *z/OS Communications Server: SNA Network Implementation Guide*.
- IPCONFIG DYNAMICXCF must be specified in the TCP/IP profile of each stack.

With this configuration, both same host and inter-host communication can be performed using Dynamic XCF.

In order to generate definitions for two TCP/IP stacks that reside on the same MVS host, you are required to specify IPCONFIG DYNAMICXCF in the TCP/IP profile of each stack.

At initialization, each TCP/IP stack configured for XCF joins a well-known XCF group. When other stacks in the group discover the new stack, the definitions are created automatically, the links are activated, and the remote IP address for each link is added to the routing table. After the remote IP address has been added, IP traffic proceeds as usual.

In VTAM, you must activate the XCF major node. You can do this using the start option XCFINIT=YES. If dynamically defined XCF definitions have been created for another VTAM in the sysplex that has since stopped and restarted with a different CPName, Dynamic XCF recognizes this situation and automatically modifies existing definitions to accommodate the CPName change. If the XCF major node is inactive when TCP/IP is started and the XCF major is not activated until after TCP/IP has finished initialization, TCP/IP will not generate any dynamic definitions for other TCP/IP hosts already started in the sysplex until either:

- A new TCP/IP host is detected
- A profile related operator command is issued (such as VARY TCPIP,,OBEYFILE, or a START or STOP command)

To request dynamics for XCF or same host connections, enter the following in the IPCONFIG statement:

```
DYNAMICXCF IPAddress SubnetMask CostMetric
```

If TCP/IP detects another instance of TCP/IP on the same z/OS and, if no device exists with the name IUTSAMEH, and if no link exists with the name EZASAMEMVS, internal definitions equivalent to the following are created:

DEVICE IUTSAMEH MPCPTP AUTORESTART
Device definition to obtain the most efficient stack-to-stack communications within the same MVS image.

LINK EZASAMEMVS MPCPTP IUTSAMEH
Link definition for the IUTSAMEH device.

HOME IPAddress EZASAMEMVS
Associates the IP address with the IUTSAMEH link.

BSDROUTINGPARMS EZASAMEMVS 65535 CostMetric SubnetMask DestIPAddress
Defines a new link to the OROUTED routing daemon.

START IUTSAMEH
Starts the IUTSAMEH device.

Note: The DestIPAddress is always 0.

If TCP/IP detects another instance of TCP/IP in the sysplex, no device with the name of the CPName of the remote VTAM exists, no iQDIO (internal Queued Direct Input/Output, or HiperSockets) connectivity between the two images exists, and no link exists with the name EZAXCFxx [where xx is the value of the MVS system symbol (SYSCLONE) for the MVS hosting the VTAM with the device name], internal definitions equivalent to the following are created:

DEVICE CPName MPCPTP AUTORESTART
Device definition to communicate with TCP/IP stacks hosted by the remote VTAM.

LINK EZAXCFnn MPCPTP CPName
Link definition for the device, where nn is the SYSCLONE value for the remote VTAM and MVS.

HOME IPAddress EZAXCFnn
Associates the IP address with the dynamic XCF link.

BSDROUTINGPARMS EZAXCFnn 55296 CostMetric SubnetMask DestIPAddress
Defines the new link to the OROUTED routing daemon.

START CPName
Starts the specified device.

Notes:

1. If EZAXCFnn is already defined as a link name or the CPName is already defined as a device name, then Dynamic XCF definitions will not be generated for discovery of another stack in the same MVS image.
2. The DestIPAddress is always zero.

If TCP/IP detects another instance of TCP/IP in the sysplex, the images reside on the same CEC, iQDIO connectivity between the two images exists, the host processor supports iQDIO and z/OS CS is properly configured, and no link exists with the name IQDIOLNKxxxxxxxx (where xxxxxxxx is the hexadecimal representation of the IP address specified on the IPCONFIG DYNAMICXCF statement), internal definitions equivalent to the following are created:

DEVICE IUTIQDIO MPCIPA AUTORESTART
Device definition to communicate with TCP/IP stacks hosted by the remote VTAM.

LINK IQDIOLNKnnnnnnnn IPAQIDIO IUTIQDIO
Link definition for the device, where nnnnnnnn is the hexadecimal representation of the IP address specified on the IPCONFIG DYNAMICXCF statement (that is, IPAddress).

HOME IPAddress IQDIOLNKnnnnnnnn
Associates the IP address with the dynamic XCF link.

BSDROUTINGPARMS IQDIOLNKnnnnnnnn 57344 CostMetric SubnetMask DestIPAddress
Defines the new link to the OROUTED routing daemon.

START IUTIQDIO
Starts the specified device.

Notes:

1. If IQDIOLNKnnnnnnnn is already defined as a link name or IUTIQDIO is already defined as a device name, Dynamic XCF definitions will not be generated for discovery of another stack in the same MVS image.
2. The DestIPAddress is always zero.

For details about these XCF-related statements, refer to *z/OS Communications Server: IP Configuration Reference*. For information about changes to NETSTAT displays of Dynamic XCF settings, refer to *z/OS Communications Server: IP System Administrator's Commands*.

IUTSAMEH

Communication Server provides internal links between TCP/IP stacks that are running within the same MVS image. This support is referred to as a Same Host (IUTSAMEH) link. If IPCONFIG DYNAMICXCF is defined, TCP/IP always creates and activates a same host (IUTSAMEH) device and link (unless a static IUTSAMEH device is already defined) even if this is the only stack on the MVS image. When TCP/IP activates the IUTSAMEH device, VTAM dynamically builds the IUTSAMEH

TRLE. The generated device name is ″IUTSAMEH″ and the generated link name is ″EZASAMEMVS″. As other stacks are brought up within the same MVS image, a host route is created to each of these stacks across the same host link. It is recommended that users do not configure a static device for IUTSAMEH (allow TCP/IP to dynamically create the device and link). Communications Server also uses the IUTSAMEH link for Enterprise Extender support.

XCF

When a subsequent stack within the sysplex is started which is not within the same MVS image, TCP/IP creates and activates an XCF device and link (unless a static XCF device is already defined). The XCF links connect using the SYSPLEX Coupling Facility (or CTC links). A new device and link are created for each corresponding stack within the sysplex. The generated device name is the (SNA/APPN) CP name of the remote VTAM. The generated link name is ″EZAXCFxx″ where xx = the two-character sysclone value. A host route across the XCF link is created when the XCF link is successfully activated.

Examples of definitions generated by Dynamic XCF

Example 1:

This configuration consists of two MVS systems (MVS1,MVS2) that are members of the same sysplex. Each MVS host has one TCP/IP stack (TCPIP1 and TCPIP2, respectively). From the syntax descriptions described above, the following information is needed to generate the dynamic definitions:

- MVS sysclone value
- VTAM CPName
- Status of XCF in VTAM
- The values specified on the IPCONFIG DYNAMICXCF keyword

Using the following user definitions:

```
MVS1:
Sysclone = A1
VTAM Cpname = VTAM1
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP1: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.1 255.255.255.248 3

MVS2:
Sysclone = B2 VTAM
Cpname = VTAM2 VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP2: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.2 255.255.255.248 2
```

After both TCPIP1 and TCPIP2 have been started, the following definitions will be generated.

TCPIP1 will generate the equivalent of these definitions:.

```
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2 MPCPTP VTAM2
HOME 9.1.1.1 EZAXCFB2
BSDROUTINGPARMS EZAXCFB2 55296 3 255.255.255.248 0
START VTAM2
```

TCPIP2 will generate:

```
DEVICE VTAM1 MPCPTP AUTORESTART
LINK EZAXCFA1 MPCPTP VTAM1
HOME 9.1.1.2 EZAXCFA1
BSDROUTINGPARMS EZAXCFA1 55296 2 255.255.255.248 0
START VTAM1
```

When an XCF link becomes active, each TCPIP will generate a route to the other TCPIP over the XCF link. In this example, when the XCF link becomes active, TCPIP1 will generate a route to TCPIP2 over the XCF link and vice versa.

Example 2:

The configuration is the same as Example 1 except a second TCP/IP stack (TCPIP1A) was added to MVS1.

Using the following user definitions:

```
MVS1:
Sysclone = A1
VTAM Cpname = VTAM1
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP1: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.1 255.255.255.248 3
TCPIP1A: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.3 255.255.255.248 0

MVS2:
Sysclone = B2
VTAM Cpname = VTAM2
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP2: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.2 255.255.255.248 2
```

After both TCPIP1 and TCPIP2 have been started, the following definitions will be generated, as in Example 1.

TCPIP1 will generate the equivalent of these definitions:

```
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2 MPCPTP VTAM2
HOME 9.1.1.1 EZAXCFB2
BSDROUTINGPARMS EZAXCFB2 55296 3 255.255.255.248 0
START VTAM2
```

TCPIP2 will generate:

```
DEVICE VTAM1 MPCPTP AUTORESTART
LINK EZAXCFA1 MPCPTP VTAM1
HOME 9.1.1.2 EZAXCFA1
BSDROUTINGPARMS EZAXCFA1 55296 2 255.255.255.248 0
START VTAM1
```

Now, TCPIP1A is started. TCPIP1 and TCPIP2 recognize that TCPIP1A has started. TCPIP1A will generate definitions for both TCPIP1 and TCPIP2. TCPIP1 will generate IUTSAMEH definitions for TCPIP1. However, TCPIP2 does not need to generate and will not generate any new definitions except for routing information for TCPIP1A. New definitions do not need to be created because the DEVICE and LINK definitions are based on the discovery of a new VTAM node in the sysplex. (The DEVICE name is the VTAM CPName.)

TCPIP1 will generate the equivalent of these definitions:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS MPCPTP IUTSAMEH
HOME 9.1.1.1 EZASAMEMVS
BSDROUTINGPARMS EZASAMEMVS 65535 3 255.255.255.248 0
START IUTSAMEH
```

When the IUTSAMEH connection becomes active, each TCPIP will generate a route to the other TCPIP over the IUTSAMEH connection.

TCPIP2 does not generate anything.

TCPIP1A will generate:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS MPCPTP IUTSAMEH
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2 MPCPTP VTAM2
HOME 9.1.1.3 EZAXCFB2
HOME 9.1.1.3 EZASAMEMVS
BSDROUTINGPARMS EZAXCFB2 55296 0 255.255.255.248 0
BSDROUTINGPARMS EZASAMEMVS 65535 0 255.255.255.248 0
START IUTSAMEH
START VTAM2
```

When the IUTSAMEH connection becomes active, each TCPIP will generate a route to the other TCPIP over the IUTSAMEH connection.

Example 3:

To continue Example 2, add another MVS host (MVS3) with a VTAM node (VTAM3) with one TCP/IP stack (TCPIP3).

```
MVS3:
Sysclone = C3
VTAM Cpname = VTAM3
VTAM has either specified XCFINIT=YES or the major node ISTLSXCF is active
TCPIP3: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.3 255.255.255.248 0
```

In this example, the previously active TCP/IP stacks will generate definitions for TCPIP3 because a new VTAM stack has become active in the sysplex. TCPIP3 will generate definitions for definitions for TCPIP1/TCPIP1A and TCPIP2.

TCPIP1 will generate the equivalent of these definitions:

```
DEVICE VTAM3 MPCPTP AUTORESTART
LINK EZAXCFC3 MPCPTP VTAM3
HOME 9.1.1.1 EZAXCFC3
BSDROUTINGPARMS EZAXCFC3 55296 3 255.255.255.248 0
START VTAM3
```

TCPIP2 will generate:

```
DEVICE VTAM3 MPCPTP AUTORESTART
LINK EZAXCFC3 MPCPTP VTAM3
HOME 9.1.1.2 EZAXCFC3
BSDROUTINGPARMS EZAXCFC3 55296 2 255.255.255.248 0
START VTAM3
```

TCPIP1A will generate:

```
DEVICE VTAM3 MPCPTP AUTORESTART
LINK EZAXCFC3 MPCPTP VTAM3
HOME 9.1.1.3 EZAXCFC3
BSDROUTINGPARMS EZAXCFC3 55296 0 255.255.255.248 0
START VTAM3
```

TCPIP3 will generate:

```
DEVICE VTAM1 MPCPTP AUTORESTART
LINK EZAXCFA1 MPCPTP VTAM1
DEVICE VTAM2 MPCPTP AUTORESTART
LINK EZAXCFB2 MPCPTP VTAM2
HOME 9.1.1.3 EZAXCFA1
HOME 9.1.1.3 EZAXCFB2
BSDROUTINGPARMS EZAXCFA1 55296 0 255.255.255.248 0
BSDROUTINGPARMS EZAXCFB2 55296 0 255.255.255.248 0
START VTAM1
START VTAM2
```

Example 4:

This example illustrates how Dynamic XCF can generate IUTSAMEH definitions without VTAM having its XCF enabled.

```
MVS1:
Sysclone = A1  (not used in this example)
VTAM Cpname = VTAM1  (not used in this example)
VTAM has XCFINIT=NO specified and has not activated the major node ISTLSXCF.
TCPIP1: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.1 255.255.255.248 3
TCPIP1A: PROFILE.TCPIP contains IPCONFIG DYNAMICXCF 9.1.1.3 255.255.255.248 0
```

TCPIP1 will generate the equivalent of these definitions:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS MPCPTP IUTSAMEH
HOME 9.1.1.1 EZASAMEMVS
BSDROUTINGPARMS EZASAMEMVS 65535 3 255.255.255.248 0
START IUTSAMEH
```

TCPIP1A will generate:

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS MPCPTP IUTSAMEH
HOME 9.1.1.3 EZASAMEMVS
BSDROUTINGPARMS EZASAMEMVS 65535 0 255.255.255.248 0
START IUTSAMEH
```

You can delete dynamically defined XCF devices and links by first stopping the devices to be deleted and then issuing a VARY TCPIP,,OBEYFILE command that contains a DELETE LINK EZAXCFxx and DELETE DEVICE. Because the HOME statement processing does not affect dynamically defined XCF HOME list entries, the HOME nn.nn.nn.nn EZAXCFxx entry is automatically deleted by DELETE LINK.

Notes:

1. The IP address of the dynamically defined devices can be changed. Because Dynamic XCF uses the same IP address for all of the dynamically defined devices, all of the dynamic devices IP addresses will be changed. An individual dynamic device cannot be changed. To change the IP addresses:
   a. Stop all of the dynamically defined devices.
   b. Issue the VARY TCPIP,,OBEYFILE command, which contains the changed IP address on the DYNAMICXCF statement.

      After they have all stopped, Dynamic XCF will change the IP address and automatically restart all of the dynamically defined devices. Dynamic XCF changes the IP address for dynamically defined XCF, IUTSAMEH links, or both in exactly the same way (with the same operational characteristics) as if you had changed the IP address for static XCF or IUTSAMEH definitions and then executed VARY TCPIP,,OBEYFILE.
2. Because the interfaces generated by Dynamic XCF use a single IP address, the output of the SIOCGIFCONF ioctl() contains multiple entries with the same IP address. If an application is using the SIOCGIFCONF output to issue bind() to all the entries returned, the application could receive EADDRINUSE on a bind() if there are multiple XCF devices defined by Dynamic XCF in the list.
3. If you want to define a static route to a link which is generated by Dynamic XCF, you must wait until the dynamic devices are started and then use the VARY TCPIP,,OBEYFILE command. The GATEWAY or BEGINROUTES statement that refers to a dynamically defined linkname must be in a separate data set from the data set used to define the dynamic devices (either initial profile data set or another OBEYFILE data set).

4. Even though the HOME, BSDROUTINGPARMS and BEGINROUTES definitions are full replacement keywords, the definitions generated by Dynamic XCF will not replace any existing definitions. Likewise, user-defined HOME, BSDROUTINGPARMS and BEGINROUTES definitions will not affect existing or future definitions generated by Dynamic XCF.

iQDIO (Internal Queued Direct Input/Output or HiperSockets)

iQDIO (Internal Queued Direct Input/Output or HiperSockets) is a new zSeries hardware feature that provides high performance internal communications between LPARs within the same CEC without the use of any additional or external hardware equipment (e.g. channel adapters, LANs, etc.). This support is also referred to as HiperSocket communications.

If the host processor supports iQDIO and CS is properly configured, CS will attempt to create XCF connectivity between same CEC LPARs using an iQDIO link. In cases in which the iQDIO link could not be activated, then TCP/IP will create a normal XCF link.

The DYNAMICXCF iQDIO device and link are dynamically built and the device is started during TCP/IP DYNAMICXCF stack initialization. The DYNAMICXCF iQDIO device and link are not (cannot be) configured by the user. The generated device name is IUTIQDIO. The generated link name is IQDIOLNK*xxxxxxxx* where *xxxxxxxx* = the character representation of the hexadecimal version of the DYNAMICXCF IP address. In general where an XCF link would normally have been used (for intra-CEC) an iQDIO link will be used.

Similar to IUTSAMEH, VTAM will dynamically build the TRLE for IUTIQDIO when the IUTIQDIO device is started. The TRLE statement is not configured (defined) by the user.

Although the DYNAMICXCF iQDIO device is not configured with TCP/IP device and link statements, and the TRLE is not defined by the user, the following steps must be taken to define the iQDIO subchannel devices and IQD CHPID:

1. Using HCD or IOCP, the system administrator must define (create the IOCDS) the iQDIO (IQD) CHPID (Channel Path ID) and subchannel devices to the applicable LPARs. In order to dynamically build the iQDIO TRLE, VTAM requires a minimum of 3 subchannel devices configured with each IQD CHPID within HCD. The maximum number of subchannel devices that VTAM will use (associate with each TRLE or MPC group) is 10. For additional details regarding configuring the iQDIO subchannel devices and IQD CHPID, refer to *z/OS HCD Planning* and Appendix D, “Using HCD” on page 757.
2. When more than one IQD CHPID is configured to a specific LPAR, VTAM start option IQDCHPID must be used to specify which specific IQD CHPID this LPAR should use. The VTAM start option controls which IQD CHPID (and related subchannel devices) VTAM selects to include in the iQDIO (IUTIQDIO) MPC Group when it is dynamically built for DYNAMICXCF iQDIO connectivity. Start option IQDCHPID controls the VTAM IQD CHPID selection for the DYNAMICXCF iQDIO device IUTIQDIO (MPC group) only. It does not control IQD CHPID selection for a user defined iQDIO (MPCIPA) device. However, a user defined iQDIO device (IQD CHPID) cannot use (conflict with) the same IQD CHPID that the DYNAMICXCF iQDIO device is currently using.

   For example, if IQD CHPID ’FE’x is currently in use by DYNAMICXCF due to one of the following:

   a. VTAM start option IQDCHPID=FE is currently specified

b. VTAM start option IQDCHPID=ANY is currently specified, but the DYNAMICXCF iQDIO device IUTIQDIO is currently using the 'FE' CHPID

then an attempt to configure and start a user defined iQDIO device IUTIQDFE will not be allowed (IQD CHPIDs conflict). This option can also be modified with a VTAM modify command. In most cases, the default setting will be sufficient. For additional details regarding this start option refer to the *z/OS Communications Server: SNA Resource Definition Reference*.

For additional details regarding iQDIO, refer to "HiperSockets concepts and connectivity" on page 130.

Workload balancing

Load balancing is the ability for a cluster to spread workload evenly (or based on some policy) to target servers comprising the cluster. Usually, this load balancing is measured by some notion of perceived load on each of the target servers. This document describes and compares three techniques that provide load balancing: DNS/WLM, Network Dispatcher, and Sysplex Distributor. Each identifies the target z/Series servers willing to receive client connections based on some specification.

By providing load balancing, clustering techniques must also provide for other system requirements in addition to the dispatching of connections. These include the ability to advertise some single systemwide image or identity so that clients can uniquely and easily identify the service. Additionally, clustering techniques should also provide for horizontal growth of the system and ease of management.

Single systemwide image

Clients connecting to a cluster should not be aware of the internal makeup of a cluster. More specifically, clients should not even be aware that the service they are requesting is actually being serviced by a collection or cluster of servers. Instead, clients must be provided with some single image identifier to be used when connecting to the service. DNS/WLM uses some specific hostname to identify a service within the cluster. In this manner, clients making requests of the service use the hostname as the single systemwide identity. In Network Dispatcher and Sysplex Distributor (SD), however, the identity is that of some IP address associated with the cluster. In the case of Sysplex Distributor, this address is a distributed Dynamic Virtual IP Address (DVIPA).

Horizontal growth

As the clients' demands on the service increase, clusters must provide a way to expand the cluster of servers to accommodate for such growing demand. Put in another way, the cluster must provide a mechanism by which to add servers without disrupting the operation of the cluster. To this end, the service is made available to clients at all times and can grow horizontally to accommodate for increased demand placed on the cluster by the clients.

Ease of management

The administrative burden associated with the cluster should not increase as we add servers to the cluster. It is desirable to use the same configurations for many systems in the cluster (sysplex). Within a sysplex, servers are homogenous, since a sysplex is comprised solely of z/Series servers. As such, many of the configurations can be shared among the different z/Series servers, thereby reducing the

administrative burden associated with the sysplex. Additionally, as the size of the cluster increases, the administrative overhead in adding systems to the cluster should be as low as possible.

DNS/WLM

The DNS solution is based on the DNS name server and the z/OS Workload Manager. This solution is only available with the BIND 4.9.3 name server and not with the BIND 9 name server. Intelligent sysplex distribution of connections is provided through cooperation between WLM and DNS. For customers who elect to place a name server in a z/OS sysplex, the name server can utilize WLM to determine the best system to service a given client request.

In general, DNS/WLM relies on the hostname to IP address resolution for the mechanism by which to distribute load among target servers. Hence, the single system image provided by DNS/WLM is that of a specific hostname. Note that the system most suitable to receive an incoming client connection is determined only at connection setup time. Once the connection is made, the system being used cannot be changed without restarting the connection.

The DNS approach works only in a sysplex environment, because the Workload Manager requires it. If the server applications are not all in the same sysplex, then there can be no single WLM policy and no meaningful coordination between WLM and DNS.

External IP workload balancing

IBM's Network Dispatcher (part of WebSphere® Edge Server) and Cisco's Multi-Node Load Balancer (MNLB) are examples of external IP workload balancing solutions. Such solutions exist outside the Sysplex, but may direct work into the Sysplex. Where DNS/WLM resolves a name to different IP addresses as a means of balancing work, external IP workload balancing solutions define a single IP address representing all instances of the server, and then balance new work requests (new TCP connection requests) among available servers. These external solutions rely on an agent in the Sysplex to deliver workload manager information for nodes with stacks on which application instances reside. All stacks hosting application instances have the same IP address defined as a hidden or loopback address. This means that normal IP routing cannot be used between the decision point and the target stack, so that either the decision point must be directly connected with the target stack - with no intervening routers - or another solution such as Generic Routing Encapsulation or other proprietary solutions must be used.

Sysplex Distributor

Sysplex Distributor is the state of the art in connection dispatching technology among z/Series IP servers. Essentially, Sysplex Distributor extends the notion of Dynamic VIPA and Automatic VIPA Takeover to allow for load distribution among target servers within the sysplex. It combines technology used with Network Dispatcher for the distribution of incoming connections with that of Dynamic VIPAs to ensure high availability of a particular service within the sysplex.

Technically speaking, the functionality of Sysplex Distributor is similar to that of Network Dispatcher in that one IP entity advertises ownership of some IP address by which a particular service is known. In this fashion, the single system image of Sysplex Distributor is also that of a special IP address. However, in the case of Sysplex Distributor, this IP address (known as the cluster address in Network Dispatcher) is called a distributed DVIPA. Further, in Sysplex Distributor, the IP

entity advertising the distributed DVIPA and dispatching connections destined for it is itself a system image within the sysplex, referred to as the distributing stack.

Like Network Dispatcher and DNS/WLM, Sysplex Distributor also makes use of Workload Manager (WLM) and its ability to gauge server load. In this paradigm, WLM informs the distributing stacks of this server load so that the distributing stack may make the most intelligent decision regarding where to send incoming connection requests. Additionally, Sysplex Distributor has the ability to specify certain policies within the Policy Agent so that it may use QoS information from target stacks in addition to WLM server load. Further, these policies can specify which target stacks are candidates for clients in particular subnetworks.

As with Network Dispatcher, connection requests are directed to the distributed stack of Sysplex Distributor. The stack selects which target server is the best candidate to receive an individual request and routes the request to it. It maintains state so that it can forward data packets associated with this connection to the correct stack. Additionally, data sent from servers within the sysplex need not travel through the distributing stack.

Sysplex Distributor also enhances the Dynamic VIPA and Automatic VIPA Takeover functions introduced in SecureWay® Communications Server for OS/390 V2R8 IP. The enhancements allow a DVIPA to move nondisruptively to another stack. That is, in the past, a DVIPA was only allowed to be active on one single stack in the sysplex. This led to potential disruptions in service when connections existed on one stack, yet the intent was to move the DVIPA to another stack. With Sysplex Distributor, the movement of DVIPAs can now occur without disrupting existing connections on the original DVIPA owning stack.

Refer to “Configuring Distributed DVIPAs — Sysplex Distributor” on page 224 for more information.

Policy interactions

The Policy Agent interacts with the Sysplex Distributor to assist with workload balancing. There will be one Policy Agent running on an LPAR regardless of how many stacks are configured. First, the Policy Agent can be configured to collect network performance statistics for applications being distributed on target stacks. These network performance statistics are then used to modify the overall WLM weight assigned to target stacks. In this way, both processor performance and application network performance are taken into account when distributing work. Second, policies established on the distributing stack can be configured to restrict the set of target stacks to be considered for any given inbound connection request. In this way, the total set of target stacks can be partitioned among different groups of users or applications requesting connections to distributed applications.

Previously, the QoS performance data was collected by the Policy Agent on the target for each DVIPA and port or application. After collecting the QoS information, the Policy Agent on the target stack pushed this information down to the stack Sysplex function which then forwarded it to the stack Sysplex function on the distributing stack. There are two significant additions to Policy Agent and Sysplex interaction:

- The Policy Agent at each target will collect information with an additional level of granularity; the QoS performance data will be collected for each service level that a target’s DVIPA port or application supports.
- The Policy Agent on the distributing stack drives the collection of this information by pulling it from the Policy Agents on the target stacks:

- The Policy Agent on the distributor opens up a TCP connection to each of the Policy Agents on the target stacks. For more information on how the Sysplex Distributor determines its targets, refer to “Configuring Distributed DVIPAs — Sysplex Distributor” on page 224.
- The Policy Agent on the distributing stack will send across a list of QoS service level names to the Policy Agent on each target.
- The Policy Agent on each target will send back a *QoS Policy Action weight fraction* for each *requested* service level that each target DVIPA port/application supports. A specific Policy Action weight fraction will not be sent unless the distributing stack's Policy Agent requests it.
- Upon receiving the *QoS Policy Action weight fraction*, the Policy Agent on the distributing stack will pass this information down to the Sysplex Distribution function on the stack. The stack Sysplex Distribution function will use this additional information when it is selecting targets for incoming connections. If it does not have a *QoS Policy Action weight fraction*, then it will use the existing weight fraction described above to make the load distribution decision instead.

How to enable Policy Agent load distribution functions:

1. Define the PolicyPerfMonitorForSDR statement in the PAGENT configuration file to enable the policy performance monitor function. This function must be active on the target and distributing stacks.
2. z/OS CS V1R2 load distribution needs to be specifically enabled for each service level; a Policy Action with the same service level name needs to be defined on each of the appropriate target stacks and also on the distributing stack for these targets. Note that it is reasonable to have a subset of key service level names defined to the distributing stack. Traffic mapping to those service level names that are defined to the distributing stack will receive z/OS CS V1R2 load distribution by service level. All other traffic will receive CS V2R10 load distribution.
3. A backup distributing stack must have the same Policy Action configuration definitions as the active distributing stack for the corresponding DVIPA targets which it is backing up, if it is desired that the Policy Action behavior stay the same when the backup distributing stack takes ownership of the DVIPA. It will also need to have the Policy Agent performance monitor function active.
4. Common PAGENT port numbers will be used by the listener (pagentQosListener) and the collector (pagentQosCollector) They are part of the /etc/services install file. If PAGENT is running on an LPAR containing a target stack, it will open a listening connection using the pagentQosListener port number. PAGENT running on an LPAR containing a distributing stack will establish a TCP connection with each PAGENT listener using the pagentQosCollector as the source port and the pagentQosListener as the destination port. The listener will fail a connect request if the source/destination port does not match the defined collector/listener port. The /etc/services file on all LPARs in the sysplex must be updated to contain these port numbers.
5. Define the two port numbers mentioned above as reserved ports for PAGENT via the PORT statement in the PROFILE.TCPIP data set.
6. Define the DYNAMICXCF parameter on the IPCONFIG statement in PROFILE.TCPIP. The PAGENT TCP connections use the XCF IP addresses.

For more information on setting up polices, refer to “Sysplex distributor policy performance monitoring configuration” on page 568, “Sysplex Distributor policy example” on page 573, or “Sysplex Distributor routing policy example” on page 580.

Connection load balancing using Sysplex Distributor in a network with CISCO routers

The IBM Sysplex Distributor (SD) function provides a workload balancing function within a parallel sysplex. The SD consists of a primary distributor stack (denoted by a Dynamic VIPA) and a set of target stacks. An inbound packet destined for that DVIPA flows through the primary distributor stack which then forwards the packet over an internal link (XCF, IUTSAMEH, or IQDIO) to the selected target stack.

The Cisco Multi-Node Load Balancer (MNLB) provides a workload balancing function which distributes traffic through Cisco routers across multiple destination TCP/IP stacks. The MNLB consists of a Service Manager (the Cisco Local Director which is denoted by a cluster IP address) and a set of Forwarding Agents (Cisco routers). For a TCP connection to the cluster IP address, the Forwarding Agent sends the SYN packet to the Service Manager, which then selects a target stack and notifies the Forwarding Agent of this decision. The Forwarding Agent then sends all future packets for that TCP connection directly to the target stack.

A solution is available to allow the customer to use a combination of the Sysplex Distributor and the MNLB to provide workload balancing.

The scope of a cluster IP address managed by Sysplex Distributor is still a single Sysplex, and integration with Cisco forwarding agents merely allows the Sysplex Distributor routing stack to be bypassed for inbound traffic. If workload balancing for a single cluster IP address across nodes in multiple clusters (Sysplexes) is desired, MNLB using Cisco Local Director as the service manager will continue to be used. Sysplex Distributor will continue to advertise network ownership of the cluster IP address with any attached routing daemon so that Sysplex Distributor appearance and behavior toward the attached routing network is unchanged except for its new relationship with Cisco forwarding agents.

This solution allows the choice of providing the workload distribution inside the sysplex, outside the sysplex, or a combination of both.

Setting up Sysplex Distributor to be the service manager for Cisco's MNLB

1. The Cisco router must be configured as a forwarding agent. The ip casa control address (which is NOT the interface address to the forwarding agent) must be advertised by the Cisco routing daemons. This is not automatically done by Cisco and must be enabled by a Cisco command. For more information on the commands, refer to Cisco's online documentation at:

   `http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t5/ipclus.htm`
2. Specify the SERVICEMGR keyword on the VIPADEFine statement in the TCPIP profile.
3. Specify the VIPASMparms statement in the TCPIP profile. Specify the same multicast group and UDP port on the VIPASMparms statement in the TCPIP profile as are configured in the MNLB.
4. Optionally, use MD5 authentication:

   Specify the same password (MD5 key) on the VIPASMparms statement in the TCPIP profile as is configured on the Cisco routers which will communicate with the Sysplex Distributor. If a password is specified, then the Sysplex Distributor will perform MD5 authentication for all communications with the Cisco Forwarding Agents. For more information on MD5 authentication, refer to RFC 1321.
5. If using the Cisco MNLB in a configuration where there is an OSA adapter between a Cisco router and the destination TCP/IP stacks such that multiple

stacks are sharing the OSA, then configure GRE tunnels on the Cisco routers. Refer to the Cisco router publications located at http://www.cisco.com/univercd/cc/td/doc/product/core/index.htm for more detailed information.

6. If using the Cisco MNLB in a mixed environment where V2R10 targets exist, routing must be configured so that the selected route from the R10 target to the routing stack is not through the Cisco MNLB service manager for this distributed DVIPA.
7. Special consideration must be made for each target stack that will receive data from an OSA that is not shared with the distributor stack. Connection load balanced IP packets routed to target stacks that do not use GRE tunnels will arrive with a destination address of the dynamic VIPA address. Only the OSA associated with the distributor stack is aware of the dynamic VIPA address. If the OSA is not the primary router, it will discard the IP packet. In this case, you must either configure GRE tunnels on the Cisco router or configure the OSA to be the default router. For more detailed information about GRE, refer to the Cisco router publications located at http://www.cisco.com/univercd/cc/td/doc/product/core/index.htm. To configure an OSA in LCS mode as the default router, use OSA/SF. To configure an OSA-Express in QDIO mode as the default router, specify PRIROUTER on the DEVICE statement. For more information on the DEVICE statement, refer to *z/OS Communications Server: IP Configuration Reference*.
8. Verification:

   The Netstat VIPADCFG/-F report may be used to verify the configuration. Refer to the *z/OS Communications Server: IP System Administrator's Commands* for more information on this command.

   Cisco's ***show ip casa*** commands may be used to display MNLB information. For more detailed information on these commands, refer to Cisco's online documentation at:

   `http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t5/ipclus.htm`

Following is a sample VIPADYNAMIC statement:

```
VIPADYNAMIC
   VIPADEFINE  MOVEABLE IMMED SERVICEMGR 255.255.255.0 197.11.221.1
   VIPASMPARMS SMMCAST 224.0.1.2 SMPORT 1637
    VIPADIST 197.11.221.1 PORT 80 20 21 23
    DESTIP 199.11.87.104
           199.11.87.105
           199.11.87.106
           199.11.87.108
           199.11.87.109
           199.11.87.110
ENDVIPADYNAMIC
```

For more information on the VIPADYNAMIC statement, refer to the *z/OS Communications Server: IP Configuration Reference*.

Part 2. Server applications

Chapter 7. Network connectivity with an SNA network

The objective of this chapter is to guide you through the steps required to implement:

- SNALINK LU0
- SNALINK LU6.2
- X.25 NPSI
- NCPROUTE

Before you configure:

Read and understand Chapter 1, “Configuration overview” on page 3. It covers important information about data set naming and search sequences.

SNALINK LU0 environment

SNALINK allows TCP/IP to send and receive packets using SNA sessions instead of dedicating physical network hardware (such as a channel-to-channel adapter or channel connection to a 3745/46 Communication Controller).

Prior to NCP V7R3, NCP did not support cross-channel native IP transmission of the transport PDUs associated with RIP traffic. NCP expects these PDUs to be carried in SNA frames. SNALINK is therefore still required for installations where dynamic routing is performed with the NCP (via NCPROUTE). See *z/OS Communications Server: IP Configuration Reference* for more information.

SNALINK allows an installation to multiplex SNA and IP traffic over the same I/O subchannels, rather than requiring separate subchannels dedicated to VTAM and TCP/IP. While such multiplexing capability may be desirable at some installations, the native TCP/IP CTC and 3745/46 device drivers will likely outperform SNALINK connections. Interaction with the SNALINK address space is very CPU-intensive, and is not required with the native TCP/IP CTC and 3745/46 device drivers. (See the *z/OS Communications Server: IP Configuration Reference* for configuration information.) It is therefore important to weigh the multiplexing capability that SNALINK provides against its performance cost, in determining whether to use SNALINK or the native TCP/IP CTC or 3745/46 device drivers.

Understanding the SNALINK environment

The SNALINK environment interfaces between the TCP/IP environment’s SNAIUCV driver and the customer’s SNA network. SNALINK communicates with one or more instances of SNALINK at remote nodes, using the SNA LU type 0 protocol. See Figure 38 on page 270 for a description of the SNALINK environment interfaces.

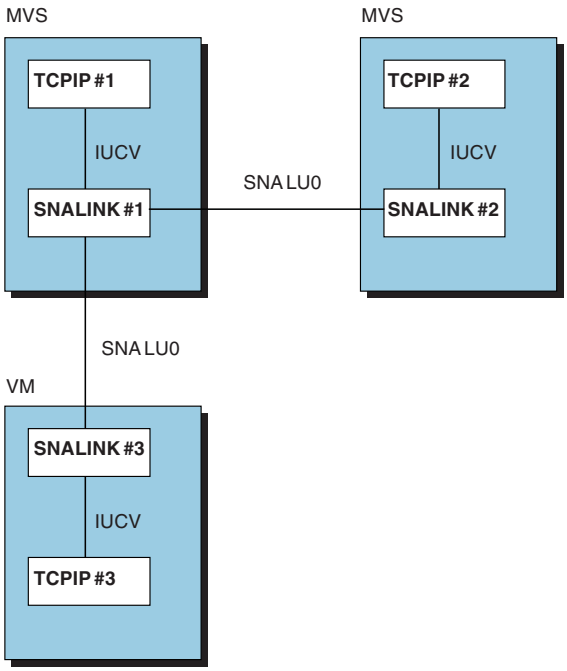


Figure 38. SNALINK environment interfaces

Each SNALINK environment can communicate with up to 9999 SNALINKs simultaneously. The number of connections is determined by the parameters you pass to the SNALINK cataloged procedure. The default is 6 sessions running in dual mode for a total of 3 SNALINKs.

- When operating in single mode, SNALINK opens one full duplex session.
- When operating in dual mode, SNALINK opens two System Network Architecture (SNA) sessions for each remote logical unit (LU) with which it communicates, one for sending and one for receiving.

Configuring SNALINK LU0

Steps to configure SNALINK LU0:

1. Specify configuration statements in *hlq*.PROFILE.TCPIP.
2. Update the SNALINK cataloged procedure.
3. Define the SNALINK application to VTAM.

Step 1: Specify configuration statements in hlq.PROFILE.TCPIP

The following sections describe the changes you must make to your TCPIP address space configuration data set (*hlq*.PROFILE.TCPIP).

Defining SNA DLC links: SNA DLC links are point-to-point and require DEVICE and LINK statements in the configuration data set. The DLC link constitutes a separate network, even though it includes only two hosts. To define a link, each host to which the DLC link is attached requires:

- A pair of SNA LU0 DEVICE and LINK statements
- A HOME statement
- A BSDROUTINGPARMS or GATEWAY or BEGINROUTES statement

SNA DLC links are defined in one of two ways:

- By unique network or subnetwork numbers, if the hosts to which they connect are not attached to other networks.

- By the IP address of the hosts to which they connect, if the hosts are attached to other networks.

You usually have to assign a unique network or subnetwork number to the SNALINK. If the link connects 2 hosts that also have other networks attached to them, the DLC link does not need its own subnetwork number. Figure 39 illustrates how to define an SNA DLC link if the 2 hosts are connected to other networks in the following way:

- Host A and Host B are connected by SNA DLC
- Host A is also connected to a token ring, 193.1.1
- Host B is also connected to a token ring, 193.1.2
- Host A's home address on its token ring is 193.1.1.1
- Host B's home address on its token ring is 193.1.2.1

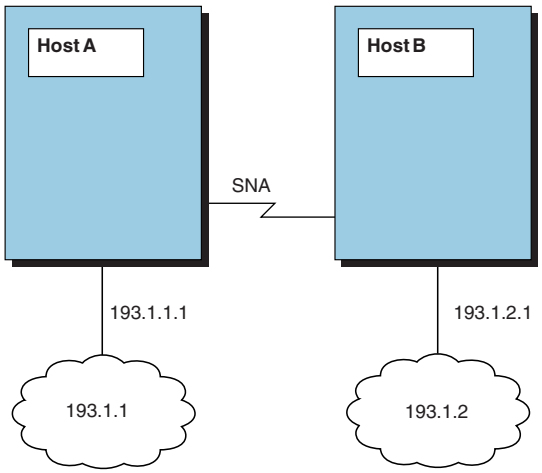


Figure 39. SNA DLC link

Host A's *hlq*.PROFILE.TCPIP could contain:

```
DEVICE  LCS1 LCS BA0
LINK  TR1  IBMTR 0 LCS1
DEVICE SNALU0 SNAIUCV SNALINK LU000000 SNALINKA
LINK SNAIUCV1 SAMEHOST  1 SNALU0
HOME
    193.1.1.1  TR1
    193.1.1.2  SNAIUCV1

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    193.1.1.0      =          TR1         2000        0
    193.1.2.0      =          SNAIUCV1    2000        0
```

Host B's *hlq*.PROFILE.TCPIP could contain:

```
DEVICE LCS2 LCS BE0
LINK  TR1 IBMTR 0 LCS2
DEVICE SNALU0 SNAIUCV SNALINK LU000001 SNALINKA
LINK SNAIUCV1 SAMEHOST  1 SNALU0
HOME
    193.1.2.1  TR1
    193.1.2.2  SNAIUCV1

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    193.1.2.0      =          TR1         2000        0
    193.1.1.0      =          SNAIUCV1    2000        0
```

Notes:

1. The *lu_name* must be different on each host. In the example, the *lu_name* for Host A is LU000000. The *lu_name* for Host B is LU000001.
2. In the example, the *lu_name* is the remote or partner LU.

Hosts A and B are addressed by their token-ring home addresses, even if the packets reach them through the SNA DLC link.

If Host B had no other network attached to it, you would have to assign a separate subnetwork number to the SNA DLC link. Even in this case, Host A does not need a separate home address for its SNA link, because it can be addressed by its token-ring home address. Host B's only home address is the home address for the SNA link.

Note: If you plan to run a network-monitoring protocol that requires each subnet to have its own subnet number, you can assign a separate subnet network number to the DLC link.

Defining NCPROUTE and 3745 LAN attachments: If your TCP/IP configuration supports NCPROUTE or 3745 Communications Controller Ethernet or token-ring links, you must do the following:

- Match the *lu_name* on the DEVICE statement to the LU statement in NCST section of your NCP generation.

  The following example shows the LU name A04TOLU1 defined in the *hlq*.PROFILE.TCPIP DEVICE statements and in the NCP generation.

```
DEVICE  SNA1LINK SNAIUCV SNALINK A04TOLU1 SNAL1STC
LINK  SNALINK SAMEHOST 1 SNA1LINK

HOME
    9.67.116.66  SNALINK

GATEWAY
;  Network         First hop  Link    Packet size    Subnet mask
    9.67.116.65   =          SNALINK     2000        HOST

START SNA1LINK
```

```
*************************************************************************
*               NCST IP INTERFACES**
*************************************************************************
A04NCSTG GROUP NCST=IP,LNCTL=SDLC,VIRTUAL=YES
A04NCSTL LINE LINEFVT=CXSXFVT,PUFVT=CXSXFVT,LUFVT=(CXSXFVT,CXSXFVT),LIN*
              ECB=CXSXLNK
A04NCSTP PU VPACING=0,PUTYPE=2,PUCB=CXSP0000S
*
A04TOLU1 LU INTFACE=(NCSTALU1,1492),REMLU=SNALKLU1,LUCB=(CXSXL0000,CXSS0*
              000),LOCADDR=1
*************************************************************************
```

- Match the remote LU name SNALKLU1 in the NCP generation to the APPLID in the SNALINK cataloged procedure parameters and in the VTAM APPL definition.

```
//SNALINK  PROC MODULE=SNALINK,TCPID='TCPV3',APPLID='SNALKLU1'
//SNALINK  EXEC PGM=&MODULE<REGION=$4096K,TIME=1440,
           PARM='&TCPID &APPLID C7 6 0003 SINGLE'
```

For additional information on configuring these links, see *z/OS Communications Server: IP Configuration Reference*.

Step 2: Update the SNALINK cataloged procedure

Update the SNALINK cataloged procedure by copying the sample in SEZAINST(SNALPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify SNALINK parameters and change the DD statements, as required. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the SNALINK cataloged procedure.

Step 3: Define the SNALINK application to VTAM

In dual mode, SNALINK opens 2 SNA sessions for each remote logical unit with which it communicates: one for sending and one for receiving. In single mode, SNALINK opens one full-duplex session.

Figure 40 is an example of a typical VTAM APPL statement for SNALINK. The application identifier (SNALKB03 in this example) must match the APPLID specified in the SNALINK cataloged procedure parameters.

```
SNALKB03  APPL ACBNAME=SNALKB03,                                    X
               AUTH=(ACQ,VPACE),                                    X
               SRBEXIT=YES,                                         X
               EAS=12,                                              X
               PARSESS=YES,                                         X
               SONSCIP=YES,                                         X
               VPACING=0
```

Figure 40. APPL statement for SNALINK

Note: ***SRBEXIT must be YES.***

Refer to *z/OS Communications Server: SNA Resource Definition Reference* more information about defining VTAM applications.

VTAM considerations:

- Each connection requires 100KB of virtual storage.
- SNALINK provides its own BIND parameters, so it does not assume or require any particular LOGMODE entries.
- The EAS value should be two times the number of maximum sessions passed to the SNALINK cataloged procedure.
- SRBEXIT=YES.
- You might have to specify pacing values (VPACING). Consult your VTAM network administrator for further details.
- For *max_ru_size*, be sure to consider the size of the TH, RH, and RU portions. If the maximum size PIU exceeds MAXRU, the NCP issues a negative response with sense 800A0000 (PIU too long). The definition used in NCP and SNALINK must be such that MAXRU is at least 29 bytes less than MAXDATA. Refer to *z/OS Communications Server: SNA Network Implementation Guide* for more information on defining the MAXDATA, MAXBFRU, and UNITSZ operands.

Stopping and starting SNALINK

If necessary, you can immediately retry a session that is waiting for the retry delay to expire by stopping and starting the SNALINK LU0 interface.

To stop SNALINK and close all connections, use the STOP command on the operator's console. For example, if SNALPROC was the name of the cataloged procedure used to start SNALINK, you enter:

```
STOP SNALPROC
```

You can also stop SNALINK with the HALT parameter on the MODIFY command. See “Controlling the SNALINK LU0 interface with the MODIFY command” on page 275.

SNALINK can be started by:

- Restarting the TCPIP address space if you have included the SNALINK procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.
- Issuing START *procname* at the command console (where *procname* is the name of the cataloged procedure used to start the SNALINK LU0 interface).

For example, to restart SNALPROC, enter

```
START SNALPROC
```

Sample console

The example in Figure 41 and the accompanying information illustrate SNALINK operation.

The line number notations in the example have been added for clarity. They do not appear in the console output.

```
Line 1              | Init complete, APPLID SNALKB03, TCPIP id TCPIPB
Line 2              | Maximum RU size is 00000600
Line 3     SNALKC04 | DLC path 00000001 pending
Line 4     SNALKC04 | Ready to accept bind from remote LU
Line 5     SNALKA04 | DLC path 00000002 pending
Line 6     SNALKA04 | Sending BIND request for SNA send session
Line 7     SNALKA04 | OPNDST   CHECK err. R15 00000004 R0 00000010 RTNCD 00000010 FDBK2 00000000
Line 8     SNALKA04 | OPNDST   sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 9     SNALKA04 | DLC path 00000002 pending
Line 10    SNALKA04 | Sending BIND request for SNA send session
Line 11    SNALKA04 | OPNDST   CHECK err. R15 00000004 R0 00000010 RTNCD 00000010 FDBK2 00000000
Line 12    SNALKA04 | OPNDST   sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 13    SNALKC04 | Received BIND request for SNA receive session
Line 14    SNALKC04 | Sending BIND request for SNA send session
Line 15    SNALKC04 | SNA receive session established
Line 16    SNALKC04 | SNA send session established
Line 17    SNALKC04 | Accepting DLC path 00000001
Line 18    SNALKA04 | DLC path 00000002 pending
Line 19    SNALKA04 | Sending BIND request for SNA send session
Line 20    SNALKA04 | SNA send session established
Line 21    SNALKA04 | Accepting DLC path 00000002
Line 22    SNALKA04 | Received BIND request for SNA receive session
Line 23    SNALKA04 | SNA receive session established
Line 24    SNALKC04 | NSEXIT CLEANUP request for receive session
Line 25    SNALKC04 | RECEIVE  CHECK err. R15 00000004 R0 0000000C RTNCD 0000000C FDBK2 0000000B
Line 26    SNALKC04 | RECEIVE  sense: SSENSEI,SSENSMI,USENSEI: 00000000
Line 27    SNALKC04 | DLC path 00000001 pending
Line 28    SNALKC04 | Ready to accept bind from remote LU
Line 29    STOP SNALINK
Line 30             | Received STOP command, shutting down
```

Figure 41. SNALINK console example

Line number	Description
Lines 1 and 2	SNALINK displays its startup information from its command line parameters, which are customized as described in *z/OS Communications Server: IP Configuration Reference*. The maximum RU size and all other values are displayed in hexadecimal.

Lines 3 and 4	The TCPIP address space, TCPIPB, issues a DLC CONNECT to establish a session with the remote LU SNALKC04. SNALKC04 is higher in the collating sequence than the local LU name SNALKB03. Consequently, SNALKB03 takes the passive role in connecting to SNALKC04, and waits for SNALKC04 to establish a session.
Lines 5 and 6	TCP/IP issues another DLC CONNECT to establish a session with SNALKA04. In this case, SNALKA04 is lower in the collating sequence. Consequently, SNALKB03 takes an active role in connecting to SNALKA04.
Lines 7 and 8	The session establishment attempt to SNALKA04 has failed, as indicated by the (nonzero) return code and the sense information printed.
Lines 9 through 12	Thirty seconds later, TCP/IP again tries to connect to SNALKA04.
Lines 13 and 14	SNALINK receives a BIND request from SNALKC04. SNALINK calls the resulting session the receive session, because it is used only to send data from SNALKC04. Now that the active end has initiated communication, SNALKB03 as the passive end, sends a BIND request to establish a send session.
Lines 15 through 17	The send and receive sessions are fully established. Establishment of the send session causes SNALINK to accept the corresponding DLC path.
Lines 18 through 23	TCP/IP again tries to connect to SNALKA04. This time it is successful (success is indicated by no nonzero return codes).
Lines 24 through 26	SNALKC04 terminates its sessions, and various error messages result.
Lines 27 and 28	Thirty seconds later, TCP/IP again tries to establish communication with SNALKC04. As in lines 13 and 14, SNALKB03 is the passive partner.
Lines 29 and 30	The operator issues a STOP SNALINK command, which causes SNALINK to stop. All DLC paths and SNA sessions are ended.

Verifying connection status using NETSTAT DEVLINKS

The DLC connect protocol between TCP/IP and SNALINK causes the status of the SNAIUCV device, reported by NETSTAT DEVLINKS, to reflect the status of the SNA sessions to the remote LU. Refer to *z/OS Communications Server: IP System Administrator's Commands* for more information on the NETSTAT command.

Controlling the SNALINK LU0 interface with the MODIFY command

Both of the following commands would pass parameters to a SNALINK LU0 address space started with a procedure named SNLK12.TCPSETUP.

```
MODIFY SNLK12.TCPSETUP,HALT

F SNLK12.TCPSETUP,PKTTRACE CLEAR *
```

TCP/IP for MVS allows the configuration of multiple DLC links to the SNALINK LU0, LU6.2, and X.25 NPSI server address spaces. The PKTTRACE parameter supports this capability through the LINKNAME parameter. Multiple PKTTRACE parameters can be issued to define the scope of the tracing by identifying the tracing options applicable to multiple links.

PKTTRACE considerations:

- Parsing of the parameter halts as soon as an error is detected and the parameter is ignored.
- Parameters can appear in any order.
- The occurrence of a parameter more than once is an error. In the case of the special parameters ON, OFF, CLEAR, and LIST, the occurrence of more than one of these parameters is an error.
- The PKTTRACE parameter must be issued after the corresponding DLC connection has been accepted from TCPIP.
- Each defined link will have an associated trace profile. The trace profile stores the effective values of each of the trace options for the link. When created or reset using the CLEAR parameter, a link's trace profile is set to the default values for the trace parameters as follows:
  DESTPORT
  No checking
  FULL Tracing of the whole IP packet
  IP All IP addresses (*)
  PROT All protocols (*)
  SRCPORT
  No checking
  SUBNET
  No checking
- Multiple statements can refer to the same link either by explicitly naming the link or by defaulting to an asterisk (*), which indicates all links. When multiple statements refer to the same link, the parameters on the statements are cumulative, and parameters not specified on the second and subsequent statements are not changed. If a parameter is specified on one statement and then appears on a subsequent statement, the value associated with the last occurrence of the option is used because this is the value that is stored in the trace.

SNALINK LU6.2

The SNALINK LU6.2 cataloged procedure runs a VTAM application program called SNALNK62, which is an interface between the TCPIP address space and the SNA network. SNALNK62 uses SNA LU type 6.2 sessions to pass the TCP/IP data to or from SNALNK62 devices running on other hosts. Examples of SNALNK62 devices include an OS/2 workstation running TCP/IP for OS/2 or a host running TCP/IP for MVS.

Configuring SNALINK LU6.2

Steps to configure SNALINK LU6.2:

1. Specify DEVICE and LINK statements in *hlq*.PROFILE.TCPIP.
2. Update the SNALINK LU6.2 cataloged procedure.

3. Define the SNALINK LU6.2 application to VTAM.
4. Update the SNALINK LU6.2 configuration data set.

Step 1: Specify DEVICE and LINK statements in hlq.PROFILE.TCPIP

You must update the *hlq*.PROFILE.TCPIP data set to include a DEVICE and LINK statement for each DLC connection to be established between the main TCPIP address space and the SNALINK LU6.2 address space.

Step 2: Update the SNALINK LU6.2 cataloged procedure

Update the SNALINK LU6.2 cataloged procedure by copying the sample in SEZAINST(LU62PROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. No system parameters are required for the SNALINK LU6.2 address space.

The DD statements in the cataloged procedure should be defined as follows:

DD Name	Description
SYSTCPD	TCPIP.DATA configuration data set
LU62CFG	SNALINK LU6.2 configuration data set
SYSPRINT	Runtime diagnostic or trace output
SYSUDUMP	User abend dump output (optional)

Refer to “Resolver configuration files” on page 27 for information on data set search sequences.

Step 3: Define the SNALINK LU6.2 application to VTAM

SNALINK LU6.2 opens two SNA LU type 6.2 sessions with each destination node; one for sending and one for receiving. If a destination node supports parallel SNA LU type 6.2 sessions (PARSESS=YES), the two sessions use the same remote logical unit; otherwise, two remote logical units are used. In either case, SNALINK LU6.2 uses a single local logical unit that must support parallel sessions.

The SNALINK LU6.2 address space must be defined to VTAM as an SNA LU type 6.2 application program. The following APPL statement defines a SNALINK LU6.2 application to VTAM.

```
LU62APPL  APPL ACBNAME=LU62APPL,                                *
               PRTCT=QWERTY,                                    *
               AUTH=(ACQ,VPACE),                                *
               SRBEXIT=NO,                                      *
               EAS=12,                                          *
               PARSESS=YES,                                     *
               SONSCIP=YES,                                     *
               APPC=YES,                                        *
               DLOGMOD=LU62MODE,                                *
               VPACING=0
```

Figure 42. APPL statement for SNALINK LU6.2

Note: ***SRBEXIT must be NO.***

See *z/OS Communications Server: SNA Resource Definition Reference* for further information about defining VTAM applications.

The LOGMODE table entry specified by the APPL DLOGMOD parameter should have the following form:

```
LU62MODE MODEENT LOGMODE=LU62MODE,FMPROF=X'13',TSPROF=X'07',          *
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',           *
               RUSIZES=X'8585',ENCR=B'0000',                          *
               PSERVIC=X'060200000000000000000300'
```

See *z/OS Communications Server: SNA Customization* for more information about defining log mode tables and *z/OS Communications Server: SNA Programming* for information on PSERVIC values.

Step 4: Update the SNALINK LU6.2 configuration data set

Customize the SNALINK LU6.2 configuration data set by copying the sample provided in SEZA.INST(LU62CFG) to your system or recognized PROCLIB and modifying it to suit your local conditions. Add or change the configuration statements as required. Be sure the //LU62CFG statement in the cataloged procedure points to this data set. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about parameters.

Sample console

The example in Figure 43 shows the messages that are expected when the SNALINK LU6.2 address space is started and a network connection is established.

```
S SNAL621A
$HASP100 SNAL621A ON STCINRDR
$HASP373 SNAL621A STARTED
 1 IEF403I SNAL621A - STARTED - TIME=15.26.03
 2 EZA5927I LU62CFG : NO ERRORS DETECTED - INITIALIZATION WILL CONTINUE
 3 EZA5932I INITIALIZATION COMPLETE - APPLID: SNAL621A  TCP/IP: TCPCS
 4 EZA5935I SEND CONVERSATION ALLOCATED FOR 9.67.22.2
 5 EZA5933I LINK SNALU62L OPENED
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE SNALU621
 4 EZA5936I RECEIVE CONVERSATION ALLOCATED FOR 9.67.22.2
```

Figure 43. Sample MVS system console messages on SNALINK LU6.2 address space startup

The following list explains the MVS system console messages on SNALINK LU6.2 address space startup as shown in Figure 43.

1 The SNAL621A address space has been started.

2 The SNALINK LU6.2 configuration data set for the SNAL621A address space has been successfully parsed.

3 The SNAL621A address space displays its local VTAM application LU and the TCP/IP address space name to which it will connect.

4 The SNAL621A address space establishes a network connection through the VTAM API.

5 The SNAL621A address space establishes a DLC connection with its TCP/IP address space.

X.25 NCP Packet Switching Interface (NPSI)

The X.25 NPSI server runs a VTAM application program called XNX25IPI, which is the interface between the TCPIP address space's DLC driver and your X.25 network. XNX25IPI communicates with the X.25 NCP Packet Switching Interface in a front-end IBM 37xx Communications Controller.

Large scale X.25 network applications often require multiple physical lines to the network switch for increased capacity and reliability. You can configure the X.25 NPSI server to support multiple lines as a group, rather than individually. In this configuration, the collection of lines is assigned a single address called a *hunt group* address. Incoming X.25 calls are distributed among the lines in either rotary or traffic balancing fashion, depending on the services offered by the X.25 network provider.

For information about improving the performance of the X.25 NPSI network, see the options on the PORT statement and GATEWAY statement in the *hlq*.PROFILE.TCPIP and the explanation provided in the *TCP/IP: Performance Tuning Guide*.

Configuring X.25 NPSI

This section describes how to configure the X.25 NPSI server.

Steps to configure the X.25 NPSI server:

1. Specify X.25 configuration statements in *hlq*.PROFILE.TCPIP.
2. Update the X.25 NPSI cataloged procedure.
3. Update the X.25 NPSI server configuration data set.
4. Define the X.25 NPSI configuration.
5. Define the X.25 NPSI application to VTAM.
6. Define VTAM Switched Circuits.

If you want to run the X.25 NPSI cataloged procedure in a different domain than the X.25 NPSI communication controller, see *z/OS Communications Server: IP Configuration Reference*.

For information about operating the X.25 NPSI server with the MODIFY command, see *z/OS Communications Server: IP Configuration Reference*.

Step 1: Specify X.25 configuration statements in hlq.PROFILE.TCPIP

To configure the *hlq*.PROFILE.TCPIP data set for X.25 NPSI, include appropriate DEVICE, LINK, HOME, GATEWAY, and START statements. The following example shows the statements that would correspond with the other X.25 samples in this chapter.

```
;
DEVICE X25DEV X25NPSI TCPIPX25
LINK X25LINK SAMEHOST 1 X25DEV
;
HOME
   199.005.058.23     X25LINK
;
GATEWAY
;
; Network  First hop   Link name Packet size  Subnet mask  Subnet value
  192.005      =        X25LINK     2000       0.0.255.0    0.0.58.0
;
START X25DEV
;
```

Note: Only one DEVICE and LINK statement per TCPIPX25 address space is allowed.

Step 2: Update the X.25 NPSI cataloged procedure

Update the X.25 NPSI cataloged procedure by copying the sample provided in SEZAINST(X25PROC) to your system or recognized PROCLIB and modifying it to suit your local conditions.

Change the data set names as needed:

- Refer to “Resolver configuration files” on page 27 for data set search sequence information.
- Modify the //X25IPI DD statement to point to your X.25 configuration data set.

Step 3: Update the X.25 NPSI server configuration data set

A sample configuration data set provided in SEZAINST(X25CONF) gives examples of how to define a public network connection, a Defense Data Network connection, and private point-to-point connection to a router. Copy this sample to the data set pointed to by the //X25IPI DD statement in your X.25 NPSI cataloged procedure. Update this sample to define your X.25 connections using the statements listed in the *z/OS Communications Server: IP Configuration Reference*.

Each connection must have a LINK and at least one DEST statement. You can optionally define hunt groups, fast connects, and call handling options for each link, and global options such as trace levels, when to clear inactive connections, and the buffer size to use for IP datagrams. You can find complete syntax for each of these statements in *z/OS Communications Server: IP Configuration Reference*.

Step 4: Define the X.25 NPSI configuration

Define the X.25 NPSI configuration according to the information in *X.25 NPSI Planning and Installation*. The X.25 NPSI server supports use of the LOGAPPL operand on the X25.MCH definition in the X.25 NPSI configuration to allow automatic recovery. You can use either the Generalized Access to X.25 Transport Extension (GATE) or Dedicated Access to X.25 Transport Extension (DATE).

IBM recommends using the X.25 NPSI GATE configuration which allows sharing of an X.25 physical link and provides better error recovery. A sample is provided in SEZAINST(NPSIGATE). NPSI GATE requires that you include the OPTIONS GATE statement in the X.25 NPSI configuration data set after the LINK statement, as shown in this portion of the X25CONF sample:

```
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU024     PRIV 1               2      1024   2
Options  GATE
*
*        IP address      X.25 DTE addr       C.U.D.
*        --------------- ---------------     --------
Dest     192.5.57.2      2
```

Sites that need to use the X.25 NPSI DATE configuration can find a sample in PV SEZAINST(NPSIDATE). See *X.25 NPSI Host Programming* for information about the definitions and parameters used in these configurations.

The following example shows portions of the sample NPSI GATE configuration (NPSIGATE). Ellipses (....) indicate code that has been omitted.

```
*********************************************************************
         OPTIONS NEWDEFN=YES,USERGEN=X25NPSI
*********************************************************************
.....
.....
```

```
NPSIV32  BUILD ADDSESS=400,                                            +
               AUXADDR=800,                                            +
               ERLIMIT=16,                                             +
               NAMTAB=120,                                             +
               MAXSESS=250,                                            +
               USGTIER=5,                                              +
               BRANCH=8000,                                            +
               BFRS=104,           BUFFER SIZE TO BE GENED             +
               CATRACE=(YES,255),  CHAN.ADAPTER TRACE OPTION           +
               CSMSG=C3D9C9E340E2C9E340D4C5E2E2C1C7C540C6D6D940E2E24040+
               40C2C340E3C5D9D4C9D5C1D3,                               +
               CWALL=26,                                               +
               ENABLTO=30.0,                                           +
               ERASE=YES,                                              +
               LOADLIB=NCPLOAD,    TARGET OF FINAL LINKEDIT            +
               LTRACE=8,           LINES TRACED SIMULTANEOUSLY         +
               MAXSSCP=8,          NUMBER OF CONCURRENT SSCP'S         +
               MODEL=3745,                                             +
               VERSION=V5R2.1,                                         +
               NEWNAME=NPSITCP,    NAME OF NCP LOAD MODULE             +
               NUMHSAS=8,          HOST SA IN CONCURRENT COMMUNICATION +
               OLT=YES,            ONLINE TERMINAL TEST                +
               PWROFF=YES,                                             +
               BACKUP=500,                                             +
               SALIMIT=511,                                            +
               SLODOWN=12,         BUFFER SLOWDOWN THRESHOLD (PERCENT) +
               SUBAREA=03,                                             +
               TRACE=(YES,100),    ADDRESS TRACE OPTION IN CORE TABLE  +
               TYPGEN=NCP,                                             +
               TYPSYS=MVS,         NCP TO BE GENERATED ON MVS          +
               TWXID=(E8D6E4C3C1D3D311,C2C9C7D5C3D7C3C1D3D325),        +
               VRPOOL=30,                                              +
               TRANSFR=32,                                             +
               NETID=NETA,                                             +
               X25.USGTIER=5,                                          +
               X25.IDNUMH=01,                                          +
               X25.MCHCNT=4,                                           +
               X25.MAXPIU=64K
.....
.....
*************************************************************************
*
*                 NPSI DEFINITIONS
*
*************************************************************************
X25XXX   X25.NET CPHINDX=1,                                            +
               NETTYPE=1,                                              +
               DM=YES,                                                 +
               OUHINDX=1
         X25.VCCPT INDEX=1,                                            +
               MAXPKTL=128,                                            +
               VWINDOW=2
         X25.OUFT INDEX=1
*-----------------------------------------------------------------------*
*        Hunt group primary line 021 with fast connect                  *
*-----------------------------------------------------------------------*
HGRP01A  X25.MCH ADDRESS=21,                                           +
               FRMLGTH=131,        128 byte packet + 3 byte header     +
               PKTMODL=8,                                              +
               ANS=CONT,                                               +
               LCGDEF=(0,16),      16 logical channels in group 0      +
               MWINDOW=2,                                              +
               STATION=DTE,                                            +
               SPEED=9600,                                             +
               LCN0=NOTUSED,                                           +
               GATE=GENERAL,       GATE                                +
               LLCLIST=(LLC4),                                         +
```

```
               CONNECT=YES,        Fast connect                       +
               LOGAPPL=TCPIPX25,                                      +
               DBIT=NO,                                               +
               DIRECT=NO,                                             +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,16),                                           +
               MAXDATA=1034,       MAXDATA only with Fast connect!    +
               TYPE=SWITCHED,                                         +
               CALL=INOUT,                                            +
               OUFINDX=1,                                             +
               VCCINDX=1
*-------------------------------------------------------------------*
*        Hunt group secondary line 022 with fast connect            *
*-------------------------------------------------------------------*
HGRP01B  X25.MCH ADDRESS=22,                                          +
               FRMLGTH=131,        128 byte packet + 3 byte header    +
               PKTMODL=8,                                             +
               ANS=CONT,                                              +
               LCGDEF=(0,16),      16 logical channels in group 0     +
               MWINDOW=2,                                             +
               STATION=DTE,                                           +
               SPEED=9600,                                            +
               LCN0=NOTUSED,                                          +
               GATE=GENERAL,       GATE                               +
               LLCLIST=(LLC4),                                        +
               CONNECT=YES,        Fast connect                       +
               LOGAPPL=TCPIPX25,                                      +
               DBIT=NO,                                               +
               DIRECT=NO,                                             +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,16),                                           +
               MAXDATA=1034,       MAXDATA only with Fast connect!    +
               TYPE=SWITCHED,                                         +
               CALL=INOUT,                                            +
               OUFINDX=1,                                             +
               VCCINDX=1
*-------------------------------------------------------------------*
*        DDN line 023                                               *
*-------------------------------------------------------------------*
DTE01    X25.MCH ADDRESS=23,                                          +
               FRMLGTH=131,        128 byte packet + 3 byte header    +
               PKTMODL=8,                                             +
               ANS=CONT,                                              +
               LCGDEF=(0,16),      16 logical channels in group 0     +
               MWINDOW=2,                                             +
               STATION=DTE,                                           +
               SPEED=9600,                                            +
               LCN0=NOTUSED,                                          +
               GATE=GENERAL,       GATE                               +
               LLCLIST=(LLC4),                                        +
               LOGAPPL=TCPIPX25,                                      +
               CTCP=(00),          paired with CUD list               +
               CUD0=(CC),          incoming CUD selects CTCP          +
               DBIT=NO,                                               +
               DIRECT=NO,                                             +
               SUBADDR=NO
         X25.LCG LCGN=0
         X25.VC LCN=(1,16),                                           +
               TYPE=SWITCHED,                                         +
               CALL=INOUT,                                            +
               OUFINDX=1,                                             +
               VCCINDX=1
*-------------------------------------------------------------------*
*        Private line 024: DCE station to router                    *
*-------------------------------------------------------------------*
```

```
 DCE01    X25.MCH ADDRESS=24,       1024 byte packet + 3 byte header    +
                FRMLGTH=1027,                                           +
                PKTMODL=8,                                              +
                ANS=CONT,                                               +
                LCGDEF=(0,2),                                           +
                MWINDOW=2,                                              +
                STATION=DCE,                                            +
                SPEED=9600,                                             +
                LCN0=NOTUSED,                                           +
                GATE=GENERAL,                                           +
                LLCLIST=(LLC4),                                         +
                CTCP=(00),          paired with CUD list                +
                CUD0=(CC),          incoming CUD selects CTCP           +
                DBIT=NO,                                                +
                DIRECT=NO,                                              +
                SUBADDR=NO
          X25.LCG LCGN=0
          X25.VC LCN=(1,2),                                             +
                TYPE=SWITCHED,                                          +
                CALL=INOUT,                                             +
                OUFINDX=1,                                              +
                VCCINDX=1
          X25.END
 ************************************************************************
·...
·...
 GENEND   GENEND
```

Step 5: Define the X.25 NPSI application to VTAM

Define the X.25 NPSI VTAM application with an APPL statement in VTAMLST. Following is an example of a VTAM APPL statement for X.25 NPSI.

```
          VBUILD TYPE=APPL                                              +
 TCPIPX25 APPL  ACBNAME=TCPIPX25,                                       +
                PRTCT=TCPX25,                                           +
                AUTH=(ACQ),                                             +
                PARSESS=YES,                                            +
                EAS=20
```

Step 6: Define VTAM switched circuits

X.25 NPSI switched virtual circuits (SVCs) appear to VTAM as switched links, [1] requiring a switched circuit definition of a physical unit (PU) and logical unit (LU) for each SVC. The sample provided in SEZAINST(X25VSVC) shows the definitions of a VTAM switched circuit corresponding to the sample X.25 NPSI GATE configuration.

The definitions are associated with the SVCs by identifying numbers (IDNUMs) created automatically during X.25 NPSI generation. The entries, in hexadecimal, run in steps of 2, by default, in the opposite order of the MCH and SVC definitions in the X.25 NPSI configuration.

Notes:

1. Permanent virtual circuits (PVCs) are not supported.
2. If you specify a local version of the z/OS UNIX table with the SSCPFM operand, the table must not have an entry for message 10 (the welcome message); otherwise, the X.25 NPSI server does not operate correctly.

Following is a sample SVC configuration data set (X25VSVC):

1. Except when using fast connect, where they appear as leased lines to VTAM. For more information, see *z/OS Communications Server: IP Configuration Reference*.

```
         VBUILD TYPE=SWNET,MAXGRP=1,MAXNO=1
*--------------------------------------------------------------------*
*   Switched circuits for DDN line 023 (16 VCs, IDNUMS 006-024)      *
*                                                                    *
*   COPYRIGHT = NONE.                                                *
*--------------------------------------------------------------------*
VP023001 PU    ADDR=23,IDBLK=003,IDNUM=01024,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023001 LU    LOCADDR=0
VP023002 PU    ADDR=23,IDBLK=003,IDNUM=01022,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023002 LU    LOCADDR=0
VP023003 PU    ADDR=23,IDBLK=003,IDNUM=01020,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023003 LU    LOCADDR=0
VP023004 PU    ADDR=23,IDBLK=003,IDNUM=0101E,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023004 LU    LOCADDR=0
VP023005 PU    ADDR=23,IDBLK=003,IDNUM=0101C,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023005 LU    LOCADDR=0
VP023006 PU    ADDR=23,IDBLK=003,IDNUM=0101A,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023006 LU    LOCADDR=0
VP023007 PU    ADDR=23,IDBLK=003,IDNUM=01018,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023007 LU    LOCADDR=0
VP023008 PU    ADDR=23,IDBLK=003,IDNUM=01016,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023008 LU    LOCADDR=0
VP023009 PU    ADDR=23,IDBLK=003,IDNUM=01014,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023009 LU    LOCADDR=0
VP023010 PU    ADDR=23,IDBLK=003,IDNUM=01012,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023010 LU    LOCADDR=0
VP023011 PU    ADDR=23,IDBLK=003,IDNUM=01010,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023011 LU    LOCADDR=0
VP023012 PU    ADDR=23,IDBLK=003,IDNUM=0100E,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023012 LU    LOCADDR=0
VP023013 PU    ADDR=23,IDBLK=003,IDNUM=0100C,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023013 LU    LOCADDR=0
VP023014 PU    ADDR=23,IDBLK=003,IDNUM=0100A,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023014 LU    LOCADDR=0
VP023015 PU    ADDR=23,IDBLK=003,IDNUM=01008,                        +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,       +
               SSCPFM=USSNTO
VL023015 LU    LOCADDR=0
VP023016 PU    ADDR=23,IDBLK=003,IDNUM=01006,                        +
```

```
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL023016 LU    LOCADDR=0
*-------------------------------------------------------------------*
*   Switched circuits for private line 024 (2 VCs, IDNUMS 002-004)    *
*-------------------------------------------------------------------*
VP024001 PU    ADDR=24,IDBLK=003,IDNUM=01004,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL024001 LU    LOCADDR=0
VP024002 PU    ADDR=24,IDBLK=003,IDNUM=01002,                          +
               DISCNT=(YES,F),MAXDATA=1034,MAXPATH=1,PUTYPE=1,         +
               SSCPFM=USSNTO
VL024002 LU    LOCADDR=0
```

NCPROUTE

NCPROUTE is a server that provides an alternative to using the Network Control Program (NCP) as a static host-independent IP router. NCPROUTE has the following effects:

- NCP becomes an active RIP router on a TCP/IP network.
- NCP becomes responsive to SNMP route table queries.

Notes:

1. NCPROUTE requires NCP V7R1, or later.
2. NCPROUTE requires SNALINK LU0 when using NCP V7R3 or previous.
3. SNALINK and IP over CDLC is supported for ESCON®, BCCA, and CADS channels.
4. IP over CDLC can be used instead of SNALINK when using NCP V7R4, or later.
5. If using RIP Version 2, NCPROUTE requires NCP V7R6, or later. Also, the NCP generation definition must have VSUBNETS=YES specified on the BUILD statement.
6. NCP versions V6R1 and V6R2 support static IP routing only. NCP uses these static route tables to deliver datagrams over connected TCP/IP networks. NCP V7R1 can be specified only as a host-dependent router and it requires the NCPROUTE server to function as a RIP router.
7. If using NCPROUTE with SNALINK, IP over CDLC channels, and OROUTED, you should customize the NCST interface metric on the NCP client side for the SNALINK NCST connection so the routes will be less preferred. This causes RouteD to prefer routes from the IP over CDLC interface over the ones from the SNALINK interface. To customize the interface metric, see the *interface metric* option in “Step 8: Configure the NCPROUTE gateways data set (Optional)” on page 300. Do the same for the SNALINK interface on the MVS host side by customizing the metric in the BSDROUTINGPARMS statement. RIP traffic will be carried over the IP over CDLC interface, while transport PDUs (for example, Hello, Add Route Request, Delete Route Request) will be carried over the SNALINK interface.
8. NCPROUTE does not support zero subnets.

NCPROUTE provides dynamic route table updates for one or more NCP clients that have been generated as IP routers and have NCPROUTE specified as the NCPROUTE server. NCPROUTE tables are updated periodically in the NCP client based on updates sent by the NCPROUTE server. These updates reflect dynamic changes in route states.

An NCPROUTE server at the host uses the Routing Information Protocol (RIP), described in RFC 1058 (RIP version 1) and in RFC 1723 (RIP version 2). The same routing protocols are used by the OROUTED server. NCPROUTE is implemented as a RIP server operating on an MVS host connected to a RIP client in the NCP. Together they provide the appearance to the TCP/IP network of an IP router using the RIP protocol. The same client/server pair also provides SNMP agent support for network management route table queries. RIP Versions 1 and 2 are currently supported by NCPROUTE. For a brief description of RIP (Versions 1 and 2), see Chapter 4, “Routing” on page 155.

Understanding the NCPROUTE environment

The NCPROUTE server:

- Supports multiple host-attached, link-attached, and remote link-attached NCP clients as illustrated in Figure 44
- Generates RIP datagrams for the NCP to send
- Maintains separate routing tables for each NCP client
- Generates SNMP route table responses for each NCP SNMP agent

The client NCP unit appears as an active router to other RIP routers on the network. Multiple NCP clients can connect to the same NCPROUTE server. Each NCP appears as an IP router to the rest of the network. Each NCP client must have one or more LU0 sessions established with SNALINK. One LU0 session per client is used as the primary session, with the remaining sessions serving as backups.

Figure 44 illustrates the different ways the NCPROUTE server can support NCP clients. NCP3 and NCP4 are host-attached NCP clients, NCP5 and NCP6 are link-attached NCP clients, and NCP1 and NCP2 are remote link-attached NCP clients.

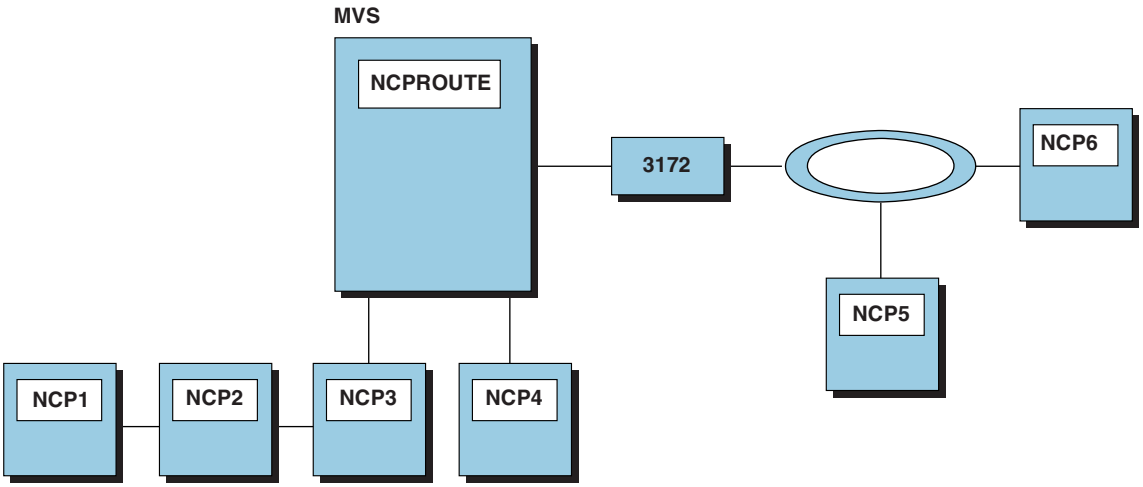


Figure 44. NCPROUTE environment

Server requirements

NCPROUTE processes RIP and SNMP datagrams addressed to all attached NCP units, generates datagrams for the NCP units, and maintains the state of each NCP unit's routing tables.

SNMP support is limited to route table queries. Queries are made to the NCP, which sends the request to the NCPROUTE server for processing.

NCPROUTE operation

An NCP's IPOWNER statement defines the controlling host and the interface this NCP client must use to reach the host. The NCP client initiates contact with NCPROUTE by sending a datagram, known as a "Hello" message, to the controlling host. It transmits this datagram on UDP port 580.

Note: The port number is generated in the NCP (using the UDPPORT keyword on the IPOWNER statement) and configured in NCPROUTE.

The "Hello" message identifies the client NCP and determines which member from the *hlq*.NCPROUTE.GATEWAYS partitioned data set to use for this NCP's route table. Any valid MVS data set name can be used for the gateways data set.

The NCP client then sends a list of its inactive links to NCPROUTE. NCPROUTE uses additional routes defined for this NCP in the NCPROUTE gateways data set, as defined in the NCPROUTE profile. It also uses the inactive links provided dynamically by the NCP to build the current route table for this NCP. The following process is repeated for each NCP that has been generated to act as a RIP router:

1. A RIP packet arrives at the NCP client from a foreign router.
2. The NCP client sends this datagram to the NCPROUTE server.
3. The NCPROUTE server processes the RIP packet.
4. The NCPROUTE server creates a RIP update for an NCP client.
5. This update is sent to the NCP client.
6. The NCP client transmits the datagram to the network.

NCPROUTE sends route table updates to each NCP client every 30 seconds. After a client has been activated, updates must be supplied over each of its interfaces every 30 seconds. The NCPROUTE server creates these updates and sends them to the NCP client along with the IP addresses of other RIP routers that the NCP client should send them to.

At the same time, adjacent RIP routers are providing periodic updates every 30 seconds to NCPROUTE. These updates are sent by the NCP client to the NCPROUTE server, where they are processed, and the results are reflected in future updates back to the NCP client.

The NCP client sends all SNMP and RIP datagrams to the NCPROUTE server for processing. The NCPROUTE server provides RIP packets and SNMP replies to the NCP client to send to their final destination.

NCPROUTE gateways:

Passive RIP route: Information about passive routes is put in NCP's and NCPROUTE's routing tables. A passive entry in NCPROUTE's routing table is used as a placeholder to prevent a route from being propagated and from being overwritten by a competing RIP route. With the exception of directly-connected passive routes, passive routes are not propagated; they are known only by this router.

Using passive routes can create routing loops, so care must be taken when creating them.

Do not define passive routes such as these:
- A to C is via B.
- B to C is via A.

Passive routes should be used when adding routes where the host or network is not running RIP. Passive routes should also be used when adding a default route, because this is the only way to prevent a route from timing out.

External RIP route: External routes are managed by other protocols, for example, the External Gateway Protocol (EGP). NCPROUTE needs to know not to interfere with these routes and not to delete them.

An external entry exists in the NCPROUTE routing table as a place holder to prevent a route from being overwritten by a competing RIP route. External routes are not propagated. NCPROUTE does not manage an external route. Therefore, NCPROUTE only knows that there is an existing route to the host or network and that is the route known by NCP.

External routes should be used when the local machine is running a non-RIP routing protocol that dynamically changes the TCP/IP routing tables. The remote machine does not need to run any routing protocol, since the only concerns are how to route traffic from the local machine to the remote machine, and how to prevent multiple routing protocols from interfering with each other.

RIP route advertising rules:

Note: RIPv1 and RIPv2 protocols are mutually exclusive; you cannot run RIPv1 and RIPv2 simultaneously.

Table 13 illustrates the differences between routing rules on the basis of RIP version.

Table 13. RIP route advertising rules

Version[2]	**Advertised destination route**[1]	**Same subnet as interface**	**Different network from interface with same subnet mask**	**Same network as interface regardless of subnet mask**	**Different network from interface**	**Same supernet as interface**	**Different supernet from interface**
RIPv1	Host	Yes[3]	Yes[3]	Yes[3]	Yes[3]		
	Subnet	No	Yes	No	No		
	Network			No	Yes		
	Supernet						
	Default				Yes[3]		

Table 13. RIP route advertising rules (continued)

Version[2]	Advertised destination route[1]	Same subnet as interface	Different network from interface with same subnet mask	Same network as interface regardless of subnet mask	Different network from interface	Same supernet as interface	Different supernet from interface
RIPv2	Host	Yes[3]	Yes[3]	Yes[3]	Yes[3]	Yes[3]	Yes[3]
	Subnet	No	Yes	Yes	Yes	Yes	Yes
	Network			No	Yes	No	Yes
	Supernet			No	Yes	No	Yes
	Default				Yes[5]		

Notes:

1. According to RIP design, route advertising relies on network-specific routes because they are the lowest common denominator. The network-specific routes consist of supernet, network, and subnet routes. The advertising of host specific routes is optional.
2. RIPv1 is the default setting for the RIP version. To set to RIPv2, specify the RIP2 parameter in NCPROUTE Profile and/or on interface options in the NCPROUTE Gateways data set.
3. The optional host specific routes are allowed to be advertised outside networks, and they are advertised in addition to the network specific routes. The option is enabled when the system -h parameter (or SUPPLY HOSTS option in NCPROUTE Gateways data set) is specified.
4. Although it is possible to advertise only the host specific routes using the RIP filters, doing so creates network unreachable problems when some routers in the network do not support the host specific routes. These routers rely on network-specific routes.
5. A default route has a network number of zero and is usually advertised over all network interfaces.
6. It does not matter whether the advertised route is VIPA or not. VIPA routes follow the same advertising rules as the non-VIPA routes.
7. Routes that are subjected to RIP filters may not be advertised at all over certain network interfaces.

NCPROUTE active gateways: Active gateways are treated as remote network interfaces. Active gateways are routers that are running RIP, but are reached through a medium that does not allow broadcasting or multicasting and is not point-to-point, for example, Hyperchannel. NCPROUTE normally requires that routers be reachable by broadcast or multicast for non-point-to-point links or by unicast addresses for point-to-point links. If the interface is neither, then an active gateway entry can add the gateway to NCPROUTE's interface list. NCPROUTE will treat the active gateway as a remote network interface. Note that the active gateway must be directly connected.

Active gateways should be used when the foreign router is reachable over a non-broadcast-capable, non-multicast-capable, and non-point-to-point network, and is directly connected to the local host.

NCPROUTE communicates with active routes by unicast transmissions to the gateway address. Routes are not added immediately to either NCPROUTE or the NCP routing table. They are added and propagated normally when route advertisements arrive from an active gateway. The sole effect of an active gateway statement is to bypass the requirement for broadcast communication on non-point-to-point links. Interfaces that are not broadcast, non-point-to-point, and are not active gateways are assumed to be loopback interfaces to the local machine. Also, while a route to an active gateway might timeout, the interface entry is never removed. If transmissions resume, then the new routes will still be available to the active gateways.

NCPROUTE gateways summary

Table 14 provides a list of NCPROUTE gateways and their characteristics.

Table 14. NCPROUTE gateways summary

	Propagate	Kernel	NCPROUTE	Timeout
Dynamic (1)	Yes	Yes	Yes	Yes
Passive	No (2)	Yes	Yes	No
External	No	No	Yes	No
Active	Yes	Yes	Yes	Yes

1 Dynamic routing is provided by NCPROUTE.

2 Except directly-connected passive routes. Directly-connected passive routes are propagated to other network interfaces for network reachability. A directly-connected passive route is one where the gateway address is one of the local interfaces in an NCP client.

RIP input/output filters

The RIP input/output filters provide routing table manipulation and routing control. The filters are provided by NCPROUTE and consist of:

- Route receiving (unconditional and conditional)
- Route noreceiving
- Route forwarding (unconditional and conditional)
- Route noforwarding
- Interface Supply switch
- Interface RIP On/Off switch
- Gateway noreceiving

For more information on these filters, see “Step 8: Configure the NCPROUTE gateways data set (Optional)” on page 300.

Configuring NCPROUTE

Steps to configure NCPROUTE:

1. Specify configuration statements in *hlq*.PROFILE.TCPIP.
2. If using SNALINK, configure VTAM and SNALINK applications.
3. If using IP over CDLC, configure IP over CDLC DEVICE and LINK statements.
4. Update the NCPROUTE cataloged procedure.
5. Update *hlq*.ETC.SERVICES.
6. Configure the host-dependent NCP clients.
7. Configure the NCPROUTE profile data set for SNMP support and specification of an NCPROUTE gateways data set (optional).
8. Configure the NCPROUTE gateways data set for each NCP client (optional).

9. If RouteD is not used, define a directly-connected host route to each NCP client.

Figure 45 shows the network addresses used in the configuration examples:

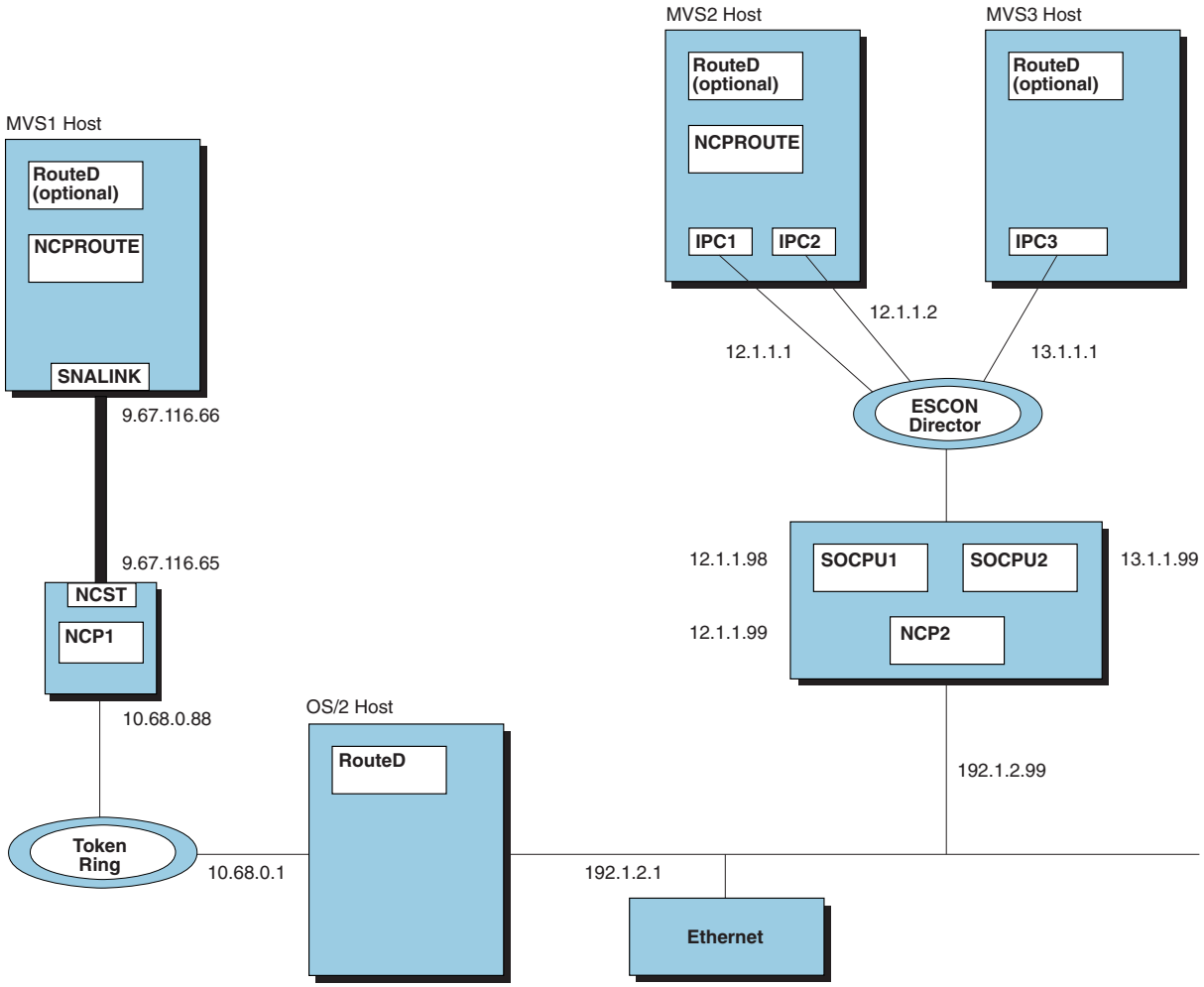


Figure 45. NCPROUTE example configuration

Step 1: Specify configuration statements in hlq.PROFILE.TCPIP

1. To have the NCPROUTE server started automatically when the TCPIP address space is started, include the name of the member containing the NCPROUTE cataloged procedure in the AUTOLOG statement in *hlq*.PROFILE.TCPIP:

```
AUTOLOG
   NCPROUT
ENDAUTOLOG
```

2. To ensure that UDP port 580 is reserved for the NCPROUTE server, also add the name of the member containing the NCPROUTE cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
   580 UDP NCPROUT
```

Note: This port number must match the one defined in the NCP generation definition (using the UDPPORT keyword on the IPOWNER statement) and assigned in *hlq*.ETC.SERVICES.

3. NCPROUTE also requires HOME and BSDROUTINGPARMS statements for the SNALINK type LU0 and IP over CDLC connections. For example, you would use this HOME and BSDROUTINGPARMS statement and, optionally, the GATEWAY statement for the configuration shown in Figure 45 on page 291:

```
MVS1:   HOME
           9.67.116.66  SNALINK
           BSDROUTINGPARMS false
             SNALINK 2000     0    255.255.240.0    9.67.116.65
           ENDBSDROUTINGPARMS
MVS2:   HOME
           12.1.1.1   IPC1
           12.1.1.2   IPC2
           BSDROUTINGPARMS false
             IPC1    1000     0    255.255.255.0   12.1.1.98
             IPC2    1000     0    255.255.255.0   12.1.1.99
           ENDBSDROUTINGPARMS
MVS3:   HOME
           13.1.1.1   IPC3
           BSDROUTINGPARMS false
             IPC3    1000     0    255.255.255.128   13.1.1.99
           ENDBSDROUTINGPARMS
```

Notes:

a. If you are not using RouteD to manage the host routes, configure static routes to the NCP client or clients in the GATEWAY statement in *hlq*.PROFILE.TCPIP. If using NCPROUTE with OMPROUTE, BSDROUTINGPARMS is required to route Transport PDUs prior to OMPROUTE activation. Since the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in both BSDROUTINGPARMS statement and the OMPROUTE configuration file. See “Step 9: Define a directly connected host route for the NCST session” on page 304 for sample definition. For more information on the GATEWAY statement, see *z/OS Communications Server: IP Configuration Reference* for each NCP client.

b. A BSDROUTINGPARMS statement is required even though RouteD is not used.

You can find a complete explanation of these configuration statements in *z/OS Communications Server: IP Configuration Reference*.

Step 2: Configure VTAM and SNALINK applications

If you are using NCP V7R3 or previous, NCPROUTE requires SNALINK type LU0 to run. To use SNALINK LU0, verify that you have configured the SNALINK LU0 interface, defined it to VTAM with a VTAM APPL definition, and included the correct DEVICE and LINK statements in *hlq*.PROFILE.TCPIP. For NCP V7R4, or later, IP over CDLC can be used instead of SNALINK.

If you are using the Cross Domain Resource (CDRSC), verify that the cross-domain resource managers are configured in VTAM.

Following is an example of an appropriate VTAM APPL definition:

```
*********************************************************************
*       SNALINK VTAM APPL DEFINITION                                *
*********************************************************************
SNALKLU1 APPL AUTH=(ACQ,VPACE),ACBNAME=SNALKLU1,EAS=12,PARSESS=YES,   *
              SONSCIP=YES,VPACING=0,SRBEXIT=YES
```

Note: The application name (the ACBNAME value, SNALKLU1, in this example) must match the REMLU interface definition in the NCP clients generation program. See the example in “Step 6: Configure the host-dependent NCP clients” on page 294 for more information.

Following is an example of corresponding DEVICE and LINK statements:

```
;
;  DEVICE AND LINK DEFINITIONS FOR SNALINK LU0
;
DEVICE SNA1LINK SNAIUCV SNALINK A04TOLU1 SNALPROC
LINK SNALINK SAMEHOST 1 SNA1LINK
;
```

Note: The LU name on the DEVICE statement (A04TOLU1 in this example) must match the LU name of the NCST interface definition in the NCP clients generation program. See the example in “Step 6: Configure the host-dependent NCP clients” on page 294 for more information.

If you want the SNALINK device to start automatically, verify that you have a START statement for this device in *hlq*.PROFILE.TCPIP. For example, `START SNA1LINK`. Otherwise, you will have to start the device manually.

Step 3: Configure the IP over CDLC DEVICE and LINK statements

For NCPROUTE, IP over CDLC can be configured along with SNALINK for NCP V7R3, or later, or it can be used to replace SNALINK for NCP V7R4, or later.

Following is an example of corresponding DEVICE and LINK statements for the configuration shown in Figure 45 on page 291 for the MVS2 host:

```
;
;  DEVICE AND LINK DEFINITIONS FOR IP OVER CDLC
;
DEVICE IPC1NCP CDLC 013 40 40 1024 1024
LINK IPC1 CDLC 0 IPC1NCP
;
DEVICE IPC2NCP CDLC 014 40 40 1024 1024
LINK IPC2 CDLC 0 IPC2NCP
```

Note: If you want a CDLC device to start automatically, verify that you have a START statement for this device in *hlq*.PROFILE.TCPIP, for example, START IPC1NCP. Otherwise, you will have to start the device manually.

Step 4: Update the NCPROUTE cataloged procedure

Update the NCPROUTE cataloged procedure by copying the sample in SEZAINST(NCPROUT) to your system or recognized PROCLIB. Specify NCPROUTE parameters and change the data set names to suit your local configuration. See Figure 46 on page 300 for an illustration of NCPROUTE data set relationships.

Step 5: Update hlq.ETC.SERVICES

NCPROUTE uses the *hlq*.ETC.SERVICES data set to determine the port number on which to run. This data set can be used to define a port number other than the reserved well-known port for NCPROUTE. This data set must exist for NCPROUTE to run.

The ETC.SERVICES data set is dynamically allocated using the standard search sequence for data set names. This data set also can be explicitly allocated in the NCPROUTE cataloged procedure using the //SERVICES DD statement.

The entries in *hlq*.ETC.SERVICES are case and column sensitive. They must be in lowercase and begin in column 1.

Add the following lines to the *hlq*.ETC.SERVICES data set:

```
ncproute  580/udp
router    520/udp
```

Note: Verify that the NCPROUTE service port number is the port being used by the NCP clients. This number should match the port number defined in the NCP generation definition using the UDPPORT keyword on the IPOWNER statement. This port number does not necessarily have to match the reserved port number for NCPROUTE on the PORT statement in *hlq*.PROFILE.TCPIP.

The reserved router service port number is 520. It is required for the NCPROUTE transport of RIP packets to NCP clients which are responsible for broadcasting the packets to other RIP routers. It cannot be overridden.

If you want to use name aliases, refer to INFO APAR II08205 for information.

Step 6: Configure the host-dependent NCP clients

You should refer to the appropriate NCP documentation for more information about defining and generating the NCP and creating route information tables.

- For more information about defining IP, refer to *NCP, SSP, and EP Resource Definition Guide*.
- For more information about the IP Dynamics function, refer to *NCP and EP Reference*.
- For more information about NCP generation definitions for IP, refer to *NCP, SSP, and EP Resource Definition Reference*.
- For more information about generating NCP as an IP router, refer to *NCP, SSP, and EP Generation and Loading Guide*.

Note: See NCPROUTE notes in on page 285.

Generating the routing information tables: To support IP dynamics, NCP's Network Definition Facility (NDF) builds a routing information table (RIT) for networks and subnetworks for use by TCP/IP at NCP generation time.

The RIT consists of routing tables that are generated from the NCP IPROUTE and IPLOCAL statements. During NCP generation, the RIT is added as a member of the NCP load library partitioned data set ncp.v7r1.ncpload. You identify the member name of ncp.v7r1.ncpload that NCPROUTE uses at execution time with the NEWNAME parameter of the BUILD statement for each NCP client generation.

Determining the gateway route table name: There is one RIT in the ncp.v7r1.ncpload data set for each NCP client this server supports. The NCPROUTE server receives the NCP name from an NCP client in the "Hello" message. This name is used as the base to determine the member name in the ncp.v7r1.ncpload partitioned data set to use for the initial RIT for this NCP client. The RIT member name in the ncp.v7r1.ncpload data set is the NEWNAME parameter of the BUILD statement for the NCP generation with a suffix of **P** added. Specify a unique name on the NEWNAME parameter of the BUILD statement for each NCP client. This name is also used as the member name if the optional gateways data set (GATEWAYS_PDS) is specified in the NCPROUTE profile. The

RIT is accessed by NCPROUTE from a //STEPLIB DD statement in the NCPROUTE cataloged procedure, LINKLST, or authorized library.

NCST session interface definition: The NCP Connectionless SNA Transport (NCST) interface is used to establish a session that can provide a connection to another IP node (NCP or z/OS) over a SNA network. Use this definition when using NCST PU interfaces to communicate with NCPROUTE using SNALINK devices with the MVS host. The NCST interface must be defined to match the SNALINK LU0 interface in VTAM so that an NCP client can establish a connection with NCPROUTE. The LU statement in the NCST interface definition tells VTAM which interface to use for the SNALINK application. The following are important keywords in this definition:

NCST
Specifies the protocol type. Must be coded as IP for internet protocol.

INTFACE
Specifies the name of the interface and the maximum transfer unit (MTU) size for the NCST session to the VTAM owner (IPOWNER).

REMLU
Specifies the name of the remote LU for the SNALINK LU0 VTAM connection. This name must match:
- The APPLID in the PROC statement of the SNALINK cataloged procedure
- The application name in the VTAM APPL definition

Note: If you define a backup NCST SNALINK session, the REMLU can specify the primary logical name for the remote LU or a different remote LU. Ensure that the MTU sizes are the same for the backup NCST sessions.

Following is an example of an NCST session interface definition:

```
*********************************************************************
*     NCST IP INTERFACES                                            *
*********************************************************************
A04NCSTG GROUP NCST=IP,LNCTL=SDLC,VIRTUAL=YES
A04NCSTL LINE LINEFVT=CXSXFVT,PUFVT=CXSXFVT,LUFVT=(CXSXFVT,CXSXFVT),   *
               LINECB=CXSXLNK
A04NCSTP PU VPACING=0,PUTYPE=2,PUCB=CXSP0000
*
A04TOLU1 LU INTFACE=(NCSTALU1,1492),REMLU=SNALKLU1,LUCB=(CXSL0000,CXSS0*
               000),LOCADDR=1
*
```

Note: The NCST LU name (A04TOLU1 in this example) must match the LU name on the SNALINK LU0 DEVICE statement in *hlq*.PROFILE.TCPIP. See the example in “Step 2: Configure VTAM and SNALINK applications” on page 292 for more information.

Channel PU interface definition: Use this definition with channel PU interfaces (ESCON, BCCA, or CADS) to communicate with NCPROUTE using IP over CDLC devices with the MVS host.

Following is an example of channel PU interface definition using the ESCON channel type:

```
*********************************************************************
*   PHYSICAL ESCON CHANNEL DEFINITIONS                              *
*********************************************************************
*
A04PSOC1 GROUP LNCTL=CA,MONLINK=NO,NPACOLL=NO,XMONLNK=YES
               SPEED=144000000,SRT=(32768,32768)
```

```
*
A04S2240 LINE  ADDRESS=2240
A04P2240 PU    ANS=CONTINUE,PUTYPE=1
*
*********************************************************************
*    LOGICAL ESCON CHANNEL DEFINITIONS                              *
*********************************************************************
*
A04PSOCB GROUP LNCTL=CA,PHYSRSC=A04P2240,NPACOLL=NO,
               DELAY=0.2,MAXPU=16,MODETAB=AMODETAB,
               DLOGMOD=INTERACT,SPEED=144000000,
               SRT=(21000,20000),PUDR=YES,
               TIMEOUT=150.0,CASDL=0.0
*
A04LSOC2 LINE  ADDRESS=NONE,HOSTLINK=1
A04L2S1  PU    ADDR=01,PUTYPE=1,ARPTAB=(10,,NOTCANON),
               INTFACE=SOCPU1
A04L2S2  PU    ADDR=02,PUTYPE=1,ARPTAB=(10,,NOTCANON),
               INTFACE=SOCPU2
```

NCP host interface definition: The IPOWNER statement in the NCP generation definition contains the TCP/IP host information and tells NCP which interface to use for NCPROUTE. The following are important keywords on the IPOWNER statement:

INTFACE
Specifies the name of the interface to the owning TCP/IP host that is running NCPROUTE.

HOSTADDR
Specifies the IP address of the owning TCP/IP. This address must match the IP address in the HOME statement in *hlq*.PROFILE.TCPIP data set for a SNALINK or IP over CDLC interface.

UDPPORT
Specifies the UDP port number for NCPROUTE. The default is 580. This port number must match the NCPROUTE service port number defined in the *hlq*.ETC.SERVICES data set. See “Step 5: Update hlq.ETC.SERVICES” on page 293 for more information.

The IPLOCAL statement in the NCP generation definition contains the NCP routing information for the local attached routes. During NCP generation, this information gets included in the Routing Information Table (RIT) which NCPROUTE uses to build the interface and routing tables. IPLOCAL routes are predefined as permanent or static to prevent modification by NCPROUTE. The following are important keywords on the IPLOCAL:

INTFACE
Specifies the name of the locally attached interface.

LADDR
Specifies the IP address of the locally attached interface.

P2PDEST
For point-to-point interfaces only. Specifies the IP address of the remote end of the point-to-point link.

PROTOCOL
Specifies the type of protocol to be used for the interface. The default is RIP which indicates that the interface is RIP-managed by NCPROUTE.

SNETMASK
Specifies the subnetwork mask for a route to a network that is subnetted.

Because RIP does not support variable subnetwork masking, this value must be equal to the subnetwork mask of the route's destination.

The IPROUTE statement in the NCP generation definition contains the NCP routing information for optional predefined routes. During NCP generation, this information gets included in RIT which NCPROUTE uses to add the routes to its routing tables. IPROUTE routes can be predefined as permanent or non-permanent for route management control by NCPROUTE. The following are important keywords on the IPROUTE statement:

INTFACE
Specifies the name of the locally attached interface for the route.

DESTADDR
Specifies the route's destination IP address.

DISP
Specifies the disposition for the route. A disposition of PERM indicates that this route is a permanent route and will not be modified by NCPROUTE. The default is NONPERM.

HOSTRT
Indicates whether this is a host route. The default is NO.

NEXTADDR
Specifies the IP address of the gateway through which the route can reach its destination. A value of 0 indicates that there is no gateway.

The following example shows typical NCP RIP router generation source statements.

```
*********************************************************************
*       IP ROUTING DEFINITIONS                                      *
*********************************************************************
*
        IPOWNER INTFACE=NCSTALU1,HOSTADDR=9.67.116.66,              *
                NUMROUTE=(100,100,100),MAXHELLO=25,UDPPORT=580
*
        IPLOCAL LADDR=9.67.116.65,INTFACE=NCSTALU1,METRIC=1,        *
                P2PDEST=9.67.116.66,PROTOCOL=RIP,SNETMASK=FFFFF000
        IPLOCAL LADDR=10.68.0.88,INTFACE=TR88,METRIC=1,             *
                SNETMASK=FFFFF000
        IPLOCAL LADDR=10.68.0.92,INTFACE=TR92,METRIC=1,             *
                SNETMASK=FFFFF000
*
        IPROUTE DESTADDR=11.0.0.1,NEXTADDR=0,INTFACE=TR88,METRIC=2, *
                DISP=PERM,HOSTRT=YES
        IPROUTE DESTADDR=12.0.0.0,NEXTADDR=13.0.0.1,INTFACE=TR92,   *
                METRIC=2,DISP=NONPERM
```

The following example shows IPOWNER and IPLOCAL statements for the ESCON channel PU interfaces in the configuration for NCP2 as shown in Figure 45 on page 291.

```
***********************************************************
*   IP ROUTING DEFINITIONS USING ESCON CHANNEL INTERFACES  *
***********************************************************
*
IPOWNER INTFACE=SOCPU1,UDPPORT=580,NUMROUTE=(110,120,130),
        HOSTADDR=12.1.1.1
*
IPLOCAL LADDR=12.1.1.98,INTFACE=SOCPU1,METRIC=1,
        P2PDEST=12.1.1.1,PROTOCOL=RIP,SUBNETMASK=FFFFFF00
*
IPLOCAL LADDR=12.1.1.99,INTFACE=SOCPU1,METRIC=1,
        P2PDEST=12.1.1.2,PROTOCOL=RIP,SUBNETMASK=FFFFFF00
```

```
*
IPLOCAL LADDR=13.1.1.99,INTFACE=SOCPU2,METRIC=1,
        P2PDEST=13.1.1.1,PROTOCOL=RIP,SUBNETMASK=FFFFFF80
```

Step 7: Configure the NCPROUTE profile data set (Optional)

To build the NCPROUTE profile, create a data set and specify its name in the //NCPRPROF DD statement in the NCPROUTE cataloged procedure. You can find a sample in SEZAINST(EZBNRPRF). Include configuration statements in this data set to define SNMP functions and to identify the NCPROUTE gateways data set. For more information, refer to *z/OS Communications Server: IP Configuration Reference*.

RIP_SUPPLY_CONTROL *supply_control*

Specifies one of the following options on a server-wide basis:

- RIP1—Unicast/Broadcast RIP Version 1 packets (Default)
- RIP2B—Unicast/Broadcast RIP Version 2 packets (Not Recommended)
- RIP2M—Unicast/Multicast/Broadcast RIP packets (Migration)
- RIP2—Unicast/Multicast RIP Version 2 packets
- NONE—Disables sending RIP packets

Note: If RIP2 is specified, the RIP Version 2 packets are multicast over multicast-capable interfaces only. No RIP packets are sent over multicast-incapable interfaces. For RIP2M, the RIP Version 2 packets are multicast over multicast-capable interfaces and RIP Version 1 packets over multicast-incapable interfaces. For RIP2B, the RIP Version 2 packets are unicast or broadcast; this option is not recommended since host route misinterpretations by adjacent routers running RIP Version 1 can occur. For this reason, RIP2B may become obsolete in a future release. For point-to-point interfaces that are non-broadcast and multicast-incapable, the RIP Version 2 packets are unicast.

RIP_RECEIVE_CONTROL *receive_control*

Specifies one of the following options on a server-wide basis:

- RIP1—Receive RIP Version 1 packets only
- RIP2—Receive RIP Version 2 packets only
- ANY—Receive any RIP Version 1 and 2 packets (Default)
- NONE—Disables receiving RIP packets

Note: If the client NCP does not support variable subnetting, the default of ANY is changed to RIP1.

RIP2_AUTHENTICATION_KEY *authentication_key*

Specifies a plain text password *authentication_key* containing up to 16 characters. The key is used on a router-wide basis and can contain mixed case and blank characters. Single quotes (') can be included as delimiters to include leading and trailing blanks. The key will be used to authenticate RIP Version 2 packets and be included in the RIP updates for authentication by adjacent routers running RIP Version 2. For maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL to RIP2. This will discard RIP1 and unauthenticated RIP2 packets. A blank key indicates that authentication is disabled. Following are examples of authentication passwords:

```
my password        (no leading or trailing blanks)
' my password '    (leading and trailing blanks)
''abc''            (single quotes part of password)
'     '            (5-character blanks)
```

SNMP_AGENT *host_name*
Specifies the host name or IP address of the host running an SNMP daemon. Only one NCPROUTE server can use a particular SNMP agent at a time.

SNMP_COMMUNITY *community_name*
Specifies a community name that SNMP applications must use to access data that the agent manages. Protect this information accordingly.

GATEWAY_PDS *dsname*
Specifies the optional partitioned data set that contains GATEWAY information for each client NCP. Quotation marks are not needed when specifying *dsname*. One member for each NCP client of this data set must be configured to match the NCP NEWNAME parameter with the **P** suffix which is the same as the NCP's RIT member name. See "Step 8: Configure the NCPROUTE gateways data set (Optional)" on page 300 for information on defining the statements necessary for the members of this data set.

Note: You can use a semicolon in column 1 to permit comments in the profile. Blank lines are also permitted.

Figure 46 on page 300 shows the relationship between the data set names specified in the NCPROUTE cataloged procedure and the NCPROUTE profile, as well as the relationship between the members of the gateways PDS and the ncpload PDS.

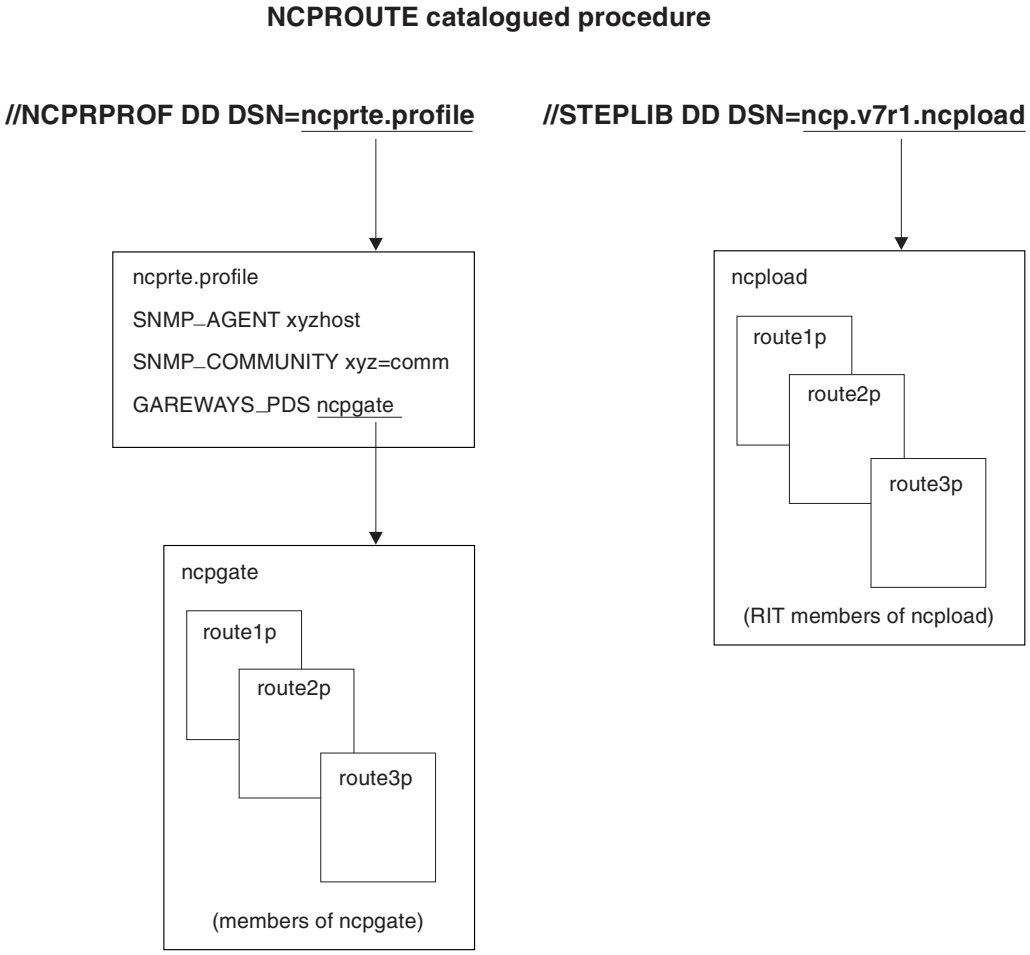


Figure 46. NCPROUTE data sets relationship

Step 8: Configure the NCPROUTE gateways data set (Optional)

The gateways data set is used to identify routes not defined in the NCP routing information table.

NCPROUTE and ROUTED require separate gateways data sets. The two servers cannot share the same data set. The NCPROUTE gateways data set is optional. However, if you use it, you must include the GATEWAY_PDS statement in the NCPROUTE profile to specify the gateway data set name. The NCPROUTE server queries the gateways data set for static routing information. It also dynamically receives routing information from the NCP client portion of this RIP router.

Allocate the gateways data set with partitioned organization (PO), a fixed block format (FB), a logical record length of 80 (LRECL), and any valid block size value for a fixed block, such as 3120.

A `passive` entry in the gateways data set is used to add a route to a part of the network that does not support RIP. An `external` entry in the gateways data set indicates a route that should never be added to the routing tables. If another RIP server offers this route to your host, the route is discarded and not added to the

routing tables. An `active` entry indicates a gateway that can only be reached through a network that does not allow or support link-level broadcasting or multicasting.

Note: The gateways data set is not related to the GATEWAY statement used in *hlq*.PROFILE.TCPIP data set.

To configure NCPROUTE, add an entry to the gateways data set for each route not defined in the NCP RIT. Use the options statement to define the characteristics of the routes in this member of the PDS.

Configuring a passive route: Figure 47 illustrates an NCPROUTE configuration using NCP as the destination hosts. In other configurations, destination hosts might not necessarily be NCPs.

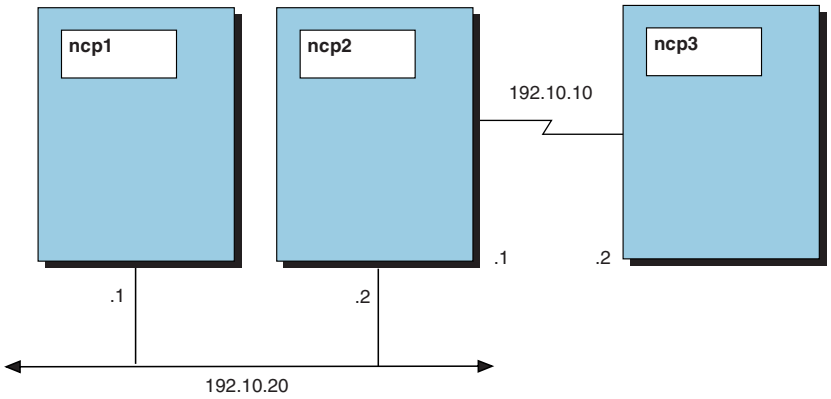


Figure 47. NCPROUTE configuration example of a passive route

Using Figure 47, assume that your NCP client `ncp1` is channel-attached to an MVS host running an NCPROUTE server. The other two NCP clients, `ncp2` and `ncp3`, are not running a RIP server. Also assume that permanent routes to `ncp2` and `ncp3` are not defined with the IPROUTE definitions in the NCP generation definition for `ncp1`. Your NCPROUTE server cannot learn a route to `ncp3`, because `ncp2` is not running a RIP server. Your NCPROUTE server sends routing updates to `ncp3` over the link to `ncp2`, but never receives a routing update from `ncp2`. After 180 seconds, your NCPROUTE server deletes the route to `ncp2`. This problem is inherent to the RIP protocol and cannot be prevented. Therefore, you need to add a `passive` route to `ncp3` in the NCPROUTE gateways data set for the NCP client `ncp1`. This is the data set defined by the GATEWAYS_PDS statement in the NCPROUTE profile.

You can use either of the following gateway statements:

```
host  ncp3         gateway  ncp2         metric 2  passive

host  192.10.10.2  gateway  192.10.20.2  metric 2  passive
```

Similarly, because `ncp2` is not running an RIP server supported by NCPROUTE, you need to add a directly-connected `passive` route as follows:

```
host  ncp2         gateway  ncp1         metric 1  passive
```

A directly-connected passive route is one where the gateway address or name is one of the local interfaces in the NCP generation.

Assume that your NCP client is now `ncp2` and is running an NCPROUTE server. `ncp1` is also running a RIP server, but `ncp3` is not. Your NCPROUTE server sends

routing information updates to ncp3 over the link to ncp3 but never receives a routing update from ncp3. After 180 seconds, your NCPROUTE server deletes the route to ncp3.

You should add a passive route to ncp3 as follows:

```
   host  ncp3        gateway  ncp2        metric 1  passive
```

ncp1 cannot reach ncp3 unless a passive routing entry is added to ncp1. For example:

```
host   ncp3    gateway  ncp2  metric  2  passive
```

or

```
host  192.10.10.2  gateway  192.10.20.2  metric  2  passive
```

Configuring an external route: Using Figure 47, assume that your NCP client ncp1 is channel-attached to an MVS host running an NCPROUTE server. The other two NCP clients, ncp2 and ncp3, are also running a RIP server. Your NCPROUTE server normally learns a route to ncp3 from ncp2, because ncp2 is running a RIP server. You might not want ncp1 to route to ncp3 for security reasons. For example, a university might want to prevent student hosts from routing to administrative hosts.

To prevent your NCPROUTE server from adding a route to ncp3, add an external route to the NCPROUTE gateways data set. This is the data set defined by the GATEWAYS_PDS statement in the NCPROUTE profile. You can use either of the following gateway statements:

```
  host  ncp3         gateway   ncp2        metric 2  external

  host  192.10.10.2  gateway  192.10.20.2  metric 2  external
```

Configuring an active gateway:

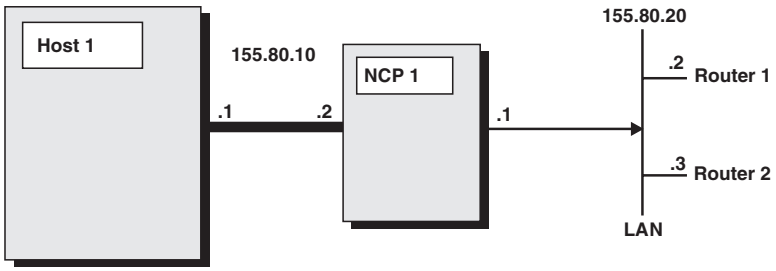


Figure 48. Configuring an active gateway

As shown in Figure 48, assume that your NCP client is ncp1, which is channel-attached to an MVS host running an NCPROUTE server and that it has a network attachment adapter that does not support link-level broadcasting or one that does not support ARP processing. Also, assume that there are routers Router1 and Router2 on the local area network. Because the IP addresses Router1 and Router2 are unknown by ncp1, they have to be manually configured in NCPROUTE for NCPROUTE to communicate with them. Configuring active gateways for Router1 and Router2 as remote network interfaces enables NCPROUTE to send RIP responses to the target addresses.

1. Specify IP addresses for each network adapter (without link-level broadcasting or ARP support) attached to the local network in the NCP client according to the NCP generation definition. For example, 155.80.20.1 is the IP address for the local network adapter attachment in `ncp1`.
2. Define active gateways for the remote routers in the NCPROUTE gateways data set specified on the GATEWAYS_PDS statement in the NCPROUTE profile:

```
active active gateway 155.80.20.2 metric 1 active
active active gateway 155.80.20.3 metric 1 active
```

NCPROUTE will use these active gateway addresses as the destination addresses to send RIP responses to the remote routers. In addition, NCPROUTE will continue to receive RIP responses from the active gateways over the NCP client.

Configuring a default route: A default route is typically used on a gateway or router to an internet, or on a gateway or router that uses another routing protocol, whose routes are not reported to other local gateways or routers.

To configure a route to a default destination, add a default route using the IPROUTE statement in the NCP generation definition. For example, if the default destination router has a gateway address 9.67.112.1, an IPROUTE statement might look like:

```
IPROUTE  DESTADDR=0.0.0.0,NEXTADDR=9.67.112.1,INTFACE=TR88,
         METRIC=1,DISP=PERM
```

An easier way would be to use the passive route definition specified in the NCPROUTE gateways data set for the NCP client. For example, the gateways entry would look like:

```
net 0.0.0.0 gateway 9.67.112.1 metric 1 passive
```

Only one default route to a destination gateway or router can be specified. For an NCP client, NCPROUTE currently does not support multiple default routes.

Configuration examples: The following example shows the contents of an NCPROUTE gateways data set containing multiple entries:

```
options default.router no trace.level 4 supply on
net testnet gateway 9.0.0.100 metric 1 passive
net 2.0.0.2 gateway 9.0.0.101 metric 2 external
host 2.0.0.3 gateway 9.0.0.102 metric 3 passive
host 2.0.0.4 gateway 9.0.0.103 metric 2 external
active active gateway 2.0.1.1 metric 1 active
```

In the second entry, the route indicates that NCPROUTE can reach network `testnet` through the gateway `9.0.0.100`, and that it is one hop away. This passive route is not broadcast to other RIP routers.

In the third entry, the route indicates that NCPROUTE can reach network `2.0.0.2` through the gateway `9.0.0.101`, and that it is two hops away. Because this route is external, NCPROUTE should not add routes for this network to the routing tables and routes received from other RIP routers for this network should not be accepted.

In the fourth entry, the route indicates that NCPROUTE can reach host `2.0.0.3` through gateway `9.0.0.102`, and that it is one hop away. This passive route is not broadcast to other RIP routers.

In the fifth entry, the route indicates that NCPROUTE can reach host `2.0.0.4` through gateway `9.0.0.103`, and that it is two hops away. Because this route is

external, NCPROUTE should not add routes for this network to the routing tables, and routes received from other RIP routers for this network should not be accepted.

The sixth entry shows an active gateway. Note that it is specified as the last entry in the data set.

Note: If a default route is to be defined to a destination gateway or router, configure a default route in this gateways data set (if the default route is not defined in a NCP client's generation definition).

Step 9: Define a directly connected host route for the NCST session

If you are not using RouteD, you need to configure a directly-connected static host route using the GATEWAY statement in *hlq*.PROFILE.TCPIP. For example, if you are using SNALINK as the host route and have the IP addresses shown in Figure 45 on page 291, the GATEWAY statement might look like this:

```
GATEWAY
; net_number first_hop link_name packet_size     subnet_mask  subnet_value
  9.67.116.65     =    SNALINK     32758           HOST
```

See *z/OS Communications Server: IP Configuration Reference* and the GATEWAY syntax information in "Step 8: Configure the NCPROUTE gateways data set (Optional)" on page 300 for more information about configuring GATEWAYS statements.

Note: The host routes on the MVS host are managed by TCP/IP as defined on the GATEWAY statement or by RouteD as defined on the BSDROUTINGPARMS statement. NCPROUTE does not manage the host routes on the MVS host. It only manages the routes on the NCP clients.

Controlling the NCPROUTE address space with the MODIFY command

You can control most of the functions of the NCPROUTE address space from the operator's console using the MODIFY command.

For information about modifying the NCPROUTE address space with the MODIFY command, refer to *z/OS Communications Server: IP System Administrator's Commands*.

Chapter 8. Accessing remote hosts using Telnet

Telnet is a terminal emulation protocol that allows end users to log on to remote host applications as though they were directly attached to that host. Telnet protocol requires that the end user has a Telnet client emulating a type of terminal the host application will understand. The client connects to a Telnet server, which communicates with the host application. The Telnet server acts as an interface between the client and host application. A PC can support several clients simultaneously, each with its own connection to any Telnet server. This chapter describes how to set up and use the following:

TN3270 Telnet server
: Provides access to z/OS VTAM SNA applications on the MVS host using Telnet TN3270E, TN3270, or linemode protocol.

z/OS UNIX Telnet server
: Provides access to z/OS UNIX shell applications on the MVS host using Telnet linemode protocol.

It is possible to use the same port for both Telnet servers. For an overview of port management, see “Port management overview” on page 55. For more specific information on the PORT BIND statement, refer to “Setting up reserved port number definitions in PROFILE.TCPIP” on page 139.

TN3270 Telnet server

The TN3270 Telnet server acts as an interface between IP and SNA networks. End users in an IP network connect to the server which is also a VTAM application. The server activates one SNA application minor node logical unit (LU) to represent each Telnet IP client. The Telnet application LU establishes a session with a VTAM host application (for example, CICS), simulating a terminal (LU0 or LU2) or a printer (LU1 or LU3). The TN3270 Telnet server runs in the TCP/IP address space as part of TCP/IP. The Telnet server is started as part of the TCP/IP startup procedure. To enable connections, the VTAM and TCP/IP configuration data sets must be modified with Telnet statements. These statements describe the Telnet LUs, a listening port, and the characteristics of that port. After TCP/IP is started, VARY and DISPLAY commands specifically related to the Telnet server can be used to alter the state of Telnet or display information about Telnet. For more information about these command sets, refer to *z/OS Communications Server: IP System Administrator's Commands*.

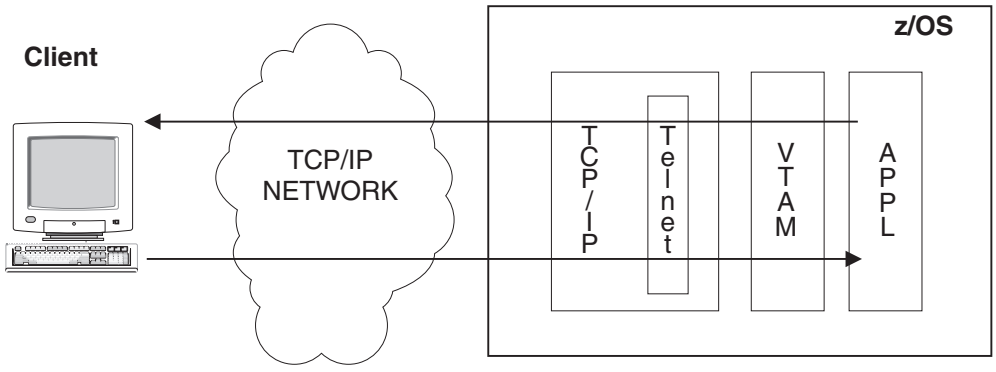


Figure 49. Telnet connectivity

Getting started

Telnet is part of the Initial Verification Procedure (IVP). With the IVP, an end user can:

- Start the TSO Telnet client
- Establish a Telnet connection to the Telnet server using loopback
- Begin a session with another TSO user ID on the same host

Establishing a new session confirms that the Telnet server is working properly. The VTAM configuration data set for Telnet is hlq.SEZAINST(IVPLU), and the Telnet configuration statements are part of the hlq.SEZAINST(SAMPPROF) TCP/IP configuration data set. For more information on IVP, refer to *z/OS Communications Server: IP Programmer's Reference*.

Starting a Telnet session

You can use the IVP sample configuration data sets to start a simple Telnet. Follow these steps to establish a TSO session over a Telnet connection:

1. Ensure the sample hlq.SEZAINST(IVPLU) member is in the concatenation of data sets specified on the VTAMLST DD statement in the procedure used to start VTAM. The LUs need to be in a connectable state. To do this, activate the major node.
2. When the TCP/IP stack is started using hlq.SEZAINST(SAMPPROF), message EZZ6003I is displayed. This indicates that Telnet is ready to accept connections.
3. To start a TSO Telnet client emulator, issue the TSO command: TELNET *ipaddr* where *ipaddr* is the TCP/IP loopback address. The command defaults to port 23. The Telnet server will use the first Telnet LU to establish a session with TSO.
4. Enter any valid TSO user ID.

This sample is designed to verify that the basic Telnet requirements are in place and functioning. Several features are not included in this sample, but you should consider them before using Telnet in production. For example, LU and application mapping, security, and connection persistence are some of the Telnet features. To learn more about Telnet capabilities and better understand TN3270, read the remainder of this chapter.

Customizing the VTAM configuration data sets

Telnet uses application LUs to represent clients that are defined in VTAM application (APPL) major nodes. Their definition members must be made available to VTAM by being in the list of data sets specified on the VTAMLST DD statement in the procedure used to start VTAM. This member contains the Telnet application LUs, and ensures that these LUs will be available for activation after VTAM is started. To automatically activate the application definition deck, include it in ATCCONxx. If multiple Telnet servers are used, for example multiple TCP/IP stacks in a sysplex distributor environment, ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail, or the cross-domain session request will fail. A sample VTAM configuration data set can be found in hlq.SEZAINST(IVPLU).

Telnet LUs, representing either terminal or printer emulators, can be defined in the VTAM configuration data set using a wildcard character instead of coding individual Telnet application LU statements. The Model Application Names function allows system administrators to code a generic APPL name with an asterisk (*) or question mark (?). Refer to *z/OS Communications Server: SNA Resource Definition Reference* for detailed information. Use * as a wildcard character to replace a

character string at the same position anywhere within the minor node name. This can produce a significant administration savings. For example, assume Telnet LUs are needed in the range of TCPABC01 through TCPABC99. The sample configuration data set has a single VTAM application definition statement with a Telnet application minor node (Telnet LU) name of TCPABC* which supports all 99 LUs.

Because Telnet server LUs do not support multiple concurrent sessions, VTAM will automatically set SESSLIM=YES for Telnet LUs defined to VTAM.

Code LOSTERM=IMMED to ensure quickest Telnet LU ACB cleanup. Without this, Telnet LUs in session with same-domain VTAM applications wait for an UNBIND response from the application.

Code EAS=1 to minimize Common Service Area (CSA) storage use. If EAS is allowed to default, excessive CSA storage will be used.

Customizing the TCP/IP configuration data sets

Telnet configuration statements are processed during initialization of the TCP/IP address space or when using the VARY TCPIP,,OBEYFILE command. The purpose of the Telnet configuration statements is to:

1. Define connection characteristics
2. Facilitate session setup with MVS host VTAM applications
3. Assign an LU name to represent the client

Telnet configuration uses the following statement blocks:

TELNETGLOBALS/ENDTELNETGLOBALS
: An *optional* statement block containing Telnet parameter statements. The parameters define connection characteristics for all ports.

TELNETPARMS/ENDTELNETPARMS
: A *required* statement block containing Telnet parameter statements. The parameters define connection characteristics for the specified port.

PARMSGROUP/ENDPARMSGROUP
: An *optional* parameter group statement within BEGINVTAM containing Telnet parameter statements. The parameters define connection characteristics for the mapped clients.

BEGINVTAM/ENDVTAM
: A *required* statement block containing Telnet mapping statements. The mapping statements define how applications and LU names are mapped (assigned) to clients.

Refer to *z/OS Communications Server: IP Configuration Reference* for exact syntax rules.

TELNETPARMS and BEGINVTAM blocks are required for each port started or modified by the VARY TCPIP,,OBEYFILE command. See “Complete profile replacement” on page 313 for details on complete profile processing. It is recommended, but not required, that you reserve the Telnet port or ports by using the stand-alone PORT *num* INTCLIEN statement in the TCP/IP startup profile. If you do not code the PORT *num* INTCLIEN statement, another application might use the port before the Telnet application can claim it.

Use a separate profile member with only Telnet statements (TELNETGLOBALS, TELNETPARMS and BEGINVTAM blocks) to keep TCP/IP configuration more

organized and allow for easy Telnet updates with the VARY TCPIP,,OBEYFILE command. To validate a Telnet profile without applying the profile, specify TESTMODE (a TELNETPARMS-only parameter). When no errors are reported, remove the TESTMODE statement and use the INCLUDE statement to add it to the TCP/IP startup profile. A sample TCP/IP configuration data set can be found in hlq.SEZAINST(SAMPPROF).

Telnet LUs can be defined to the Telnet configuration data set using several different wildcard notations instead of coding individual LU names. The DEFAULTLUS statement can be used to define terminal emulator LUs. The sample profile defines to Telnet application LUs TCPABC01 through TCPABC99 to represent clients connecting to port 23 of the stack IP address. The last two fields are defined to be numeric variables. If multiple Telnet servers are used, for example multiple TCP/IP stacks in a sysplex distributor environment, ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail, or the cross-domain session request will fail. After connecting, the DEFAULTAPPL statement maps the TSO application to the client and causes the Telnet server to immediately initiate a session between the Telnet LU and TSO. If an error occurs during session initiation, the MSG07 statement allows an error message to be sent to the client. If MSG07 is not coded, the connection is dropped. The sample profile contains additional statements that are included as comments. These statements provide examples of advanced functions. Many of these statements are installation-specific and will require modification.

Connection characteristic parameters: Telnet initially sets all connection parameters to default values. Parameters can then be changed at three levels. Each level provides a different scope of coverage. Parameters merge down as shown in Figure 50. Parameters coded in TELNETGLOBALS will be applied to all connections on all ports, unless overridden by parameters in either TELNETPARMS or PARMSGROUP. Parameters coded in TELNETPARMS will be applied to all connections on the specified port, unless overridden by parameters in PARMSGROUP. Parameters coded in PARMSGROUP will be applied to connections whose clients are mapped to that PARMSGROUP. Telnet parameters are described throughout this chapter where appropriate. For a complete list, see *z/OS Communications Server: IP Configuration Reference*.

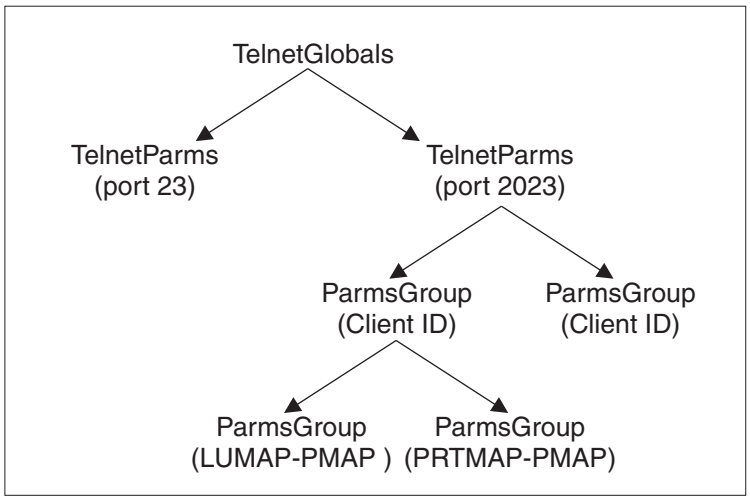


Figure 50. Telnet parameter placement

Session setup and LU assignment: There are ten mapping statements available in the BEGINVTAM block, which map ten different Objects to clients. Five are application setup related, four are LU name assignment related, and one maps connection parameters to specified Client Identifiers within the port. Shortly after a connection request is accepted, the mapping statements are used by Telnet to map, or assign, as many of the ten Objects to the client as possible. This set of Objects is used for the duration of the connection. For more detailed information, refer to "Mapping Objects to Client Identifiers" on page 325.

Managing the Telnet server

Commands

Many networking products (such as VTAM) use VARY commands to change the state of a device and DISPLAY commands to show information. The Telnet server also uses VARY and DISPLAY commands to change and monitor Telnet functions and debug problems.

Note that when a VARY TCPIP,,OBEYFILE command is issued, TELNETPARMS and BEGINVTAM blocks are both required for each port started or modified.

The following commands help manage the Telnet server. Syntax and examples can be found in *z/OS Communications Server: IP System Administrator's Commands*.

- Telnet VARY commands allow the operator to change the state of Telnet ports, enable or disable the use of certain Telnet LU names, and manage diagnostic tools. These commands include:
  - VARY TCPIP,,TELNET,QUIESCE a port to block any new connection requests but allow existing connections to continue activity.
  - VARY TCPIP,,TELNET,RESUME a port to end the QUIESCEd state and allow new connection requests.
  - VARY TCPIP,,TELNET,STOP a port to end connections on the port and close the port.
  - VARY TCPIP,,OBEYFILE a profile to start, restart or change a port by updating the Telnet profile. The VARY TCPIP,,TELNET,STOP and VARY TCPIP,,OBEYFILE commands can be used to stop a Telnet port and then restart that port or a new port without stopping the TCP/IP stack.
  - VARY TCPIP,,TELNET,ACT and VARY TCPIP,,TELNET,INACT LUs for use by the Telnet server. If an LU is already in use, the INACT command fails. Specify the name ALL to activate all inactive LUs with one command. These commands have no effect on the VTAM state of the LU.
  - VARY TCPIP,,TELNET,DEBUG,OFF turns off all debug activity that might have been turned on to debug a problem.
  - VARY TCPIP,,TELNET,ABENDTRAP sets an abend trap based on unique Telnet return codes set in Telnet modules.

  WLM registration is affected by port changes. See "WorkLoad Manager for Telnet (WLM)" on page 372 for more information.
- Telnet DISPLAY commands are discussed in "Telnet diagnostics" on page 367.

Qualified ports

In some cases all clients need to use the same port number, but the Telnet parameters need to be differentiated by destination IP address or destination linkname.

For example, two stacks are running with a Telnet server on each stack, and the two stacks are going to be merged into one stack. Currently, stack 1 has a home address of 1.1.1.1 and has Telnet running with a set of definitions for port 23. Stack 2 has a home address of 2.2.2.2 and has Telnet running with a different set of definitions for port 23. Before the stacks are merged into one, end users connect to either 1.1.1.1,port 23 or 2.2.2.2,port 23, depending on which Telnet services are desired. The sample definition statements are:

```
Stack 1

 TelnetParms
  Port 23
  Inactive 600      ; Drop after 10 minutes of no activity
 EndTelnetParms

 BeginVTAM
  Port 23
  DefaultLus TCPABC01..TCPABC49 EndDefaultLus
  DefaultAppl TSO
 EndVTAM

Stack 2

 TelnetParms
  Port 23
  Inactive 0        ; Never drop
 EndTelnetParms

 BeginVTAM
  Port 23
  DefaultLus TCPABC50..TCPABC99 EndDefaultLus
  DefaultAppl CICS
 EndVTAM
```

After the merge, both home addresses exist in a single stack. One way to keep the Telnet definitions separate would be to change the port number in one of the definition sets. For instance, the port 23 definitions associated with the old stack 2 could be changed to be port 223. The end result is one TCPIP stack and one Telnet server with port 23 and port 223, where port 23 has the definitions used in the old stack 1 and port 223 has the definitions used in the old stack 2. The definitions are still separate. However, all the end users who were connecting to 2.2.2.2 port 23 now have to change their clients to port 223. The sample definition statements would be changed to:

```
Merged Stack

 TelnetParms
  Port 23
  Inactive 600      ; Drop after 10 minutes of no activity
 EndTelnetParms

 BeginVTAM
  Port 23
  DefaultLus TCPABC01..TCPABC49 EndDefaultLus
  DefaultAppl TSO
 EndVTAM

 TelnetParms
  Port 223
  Inactive 0        ; Never drop
 EndTelnetParms

 BeginVTAM
```

```
 Port 223
 DefaultLus TCPABC50..TCPABC99 EndDefaultLus
 DefaultAppl CICS
EndVTAM
```

With port qualification, the system administrator can qualify the port number with the destination IP address or linkname to keep the Telnet services separate. In this case, the destination IP address is used. The qualified port allows the users of either old stack to connect without making any changes to their client. The sample definition statements would be changed to:

```
Merged Stack

TelnetParms
 Port 23,1.1.1.1
 Inactive 600      ; Drop after 10 minutes of no activity
EndTelnetParms

BeginVTAM
 Port 23,1.1.1.1
 DefaultLus TCPABC01..TCPABC49 EndDefaultLus
 DefaultAppl TSO
EndVTAM

TelnetParms
 Port 23,2.2.2.2
 Inactive 0        ; Never drop
EndTelnetParms

BeginVTAM
 Port 23,2.2.2.2
 DefaultLus TCPABC50..TCPABC99 EndDefaultLus
 DefaultAppl CICS
EndVTAM
```

You cannot QUIESCE, RESUME, or STOP a qualified portion of a port. If the port has several qualified port profiles, the VARY TCPIP,,QUIESCE, the VARY TCPIP,,RESUME, and the VARY TCPIP,,STOP commands affect all qualified port profiles associated with the port being quiesced, resumed, or stopped. In the example above, V TCPIP,,T,STOP,PORT=23 will stop port 23,1.1.1.1 and port 23,2.2.2.2. It is not possible to stop port 23,1.1.1.1 or port 23,2.2.2.2 individually. All display commands that allow port specification allow you to specify a qualified port. If just the port number is specified, all qualified port profiles are displayed. DBCSTRANSFORM can be active on only one port, but can be active on one, some, or all of the qualified profiles of that port.

Multiple ports

Telnet supports up to 255 ports on one TCP/IP stack. A unique TELNETPARMS block must be created for each port or qualified port. Telnet allows the use of the same BEGINVTAM block for all ports, some ports, or a unique BEGINVTAM block for each port. Both TELNETPARMS and BEGINVTAM blocks are required for each port started or modified by a VARY TCPIP,,OBEYFILE command. One or more PORT *num* TCP INTCLIEN reservation statements can be specified. The first PORT *num* TCP INTCLIEN specified will generate a default TELNETPARMS block and combine with a single BEGINVTAM block to start Telnet. It is recommended that every port be defined with explicit TELNETPARMS blocks to avoid confusion. There are several reasons more than one Telnet port or qualified port might be needed. The two most common reasons are discussed in the following sections.

Assigning a single application to a port simplifies the setup of clients on the workstation and the logon process. Workstation clients can be labeled with the

associated application name and then be set up to connect to the appropriate port or qualified port. With a client per application on the workstation, the end user can select the needed client, connect, and be immediately in session with the application defined on the DEFAULTAPPL statement in BEGINVTAM. This implementation requires a unique BEGINVTAM block for each port due to the unique DEFAULTAPPL statements. The example below shows how to set up TSO, IMS, and CICS on ports 23, 223, and 423, respectively. The same LU names are used in each BEGINVTAM block. Telnet maintains a master LU ″in-use″ registry across all ports so that the same LU name will not be used by two different ports.

```
TELNETPARMS
      PORT  23
ENDTELNETPARMS
TELNETPARMS
      PORT  223
ENDTELNETPARMS
TELNETPARMS
      PORT  423
ENDTELNETPARMS

BEGINVTAM
      PORT  23
      DEFAULTLUS TCPABC01..TCPABC99 ENDDEFAULTLUS
      DEFAULTAPPL   TSO
ENDVTAM
BEGINVTAM
      PORT  223
      DEFAULTLUS TCPABC01..TCPABC99 ENDDEFAULTLUS
      DEFAULTAPPL   IMS
ENDVTAM
BEGINVTAM
      PORT  423
      DEFAULTLUS TCPABC01..TCPABC99 ENDDEFAULTLUS
      DEFAULTAPPL   CICS
ENDVTAM
```

Assigning different security levels to different ports is an easy way to differentiate client security needs. External connections might require SSL security, while internal connections do not. Other than that difference, all other aspects of the Telnet profile can be the same. For example, external clients can connect to port 23 of a firewall that converts the request to the Telnet secure port 992. Internal clients would connect directly to the Telnet basic port 23. The statements below show how two ports allow implementation of different security levels. Note the same BEGINVTAM block is used for both ports, which can significantly reduce profile maintenance complexity. The PORT statement in BEGINVTAM links the BEGINVTAM block to the multiple TELNETPARMS blocks defined.

```
TELNETPARMS
      PORT  23
ENDTELNETPARMS
TELNETPARMS
      SECUREPORT  992
      KEYRING hfs /use/keyring/tcpcs.kdb
ENDTELNETPARMS
BEGINVTAM
      PORT  23 992
      DEFAULTLUS TCPABC01..TCPABC99 ENDDEFAULTLUS
      ALLOWAPPL   *
ENDVTAM
```

If a profile that contains a new port number is processed, it is treated as an additional port, and the VARY TCPIP,,OBEYFILE request will succeed if all

parameters for the new port are correctly specified. Existing, non-referenced ports remain active and unchanged. You can use the VARY TCPIP,,TELNET,STOP command to stop a port.

See “WorkLoad Manager for Telnet (WLM)” on page 372 for more information about multiport considerations.

Complete profile replacement

When using VARY TCPIP,,OBEYFILE to update the Telnet configuration, new profile statements completely replace the profile statements that were in use before the update. For a successful port update, both TELNETPARMS and BEGINVTAM blocks are required for each port started or modified. The updates are *not* cumulative from the previous profile. If only one change is needed in the new profile, change the old profile or copy the profile to another data set member and make the change. After VARY TCPIP,,OBEYFILE processing, the new profile is labeled the CURRent profile, and the replaced profile becomes profile 0001. If another update is done, the new update becomes the current profile and the replaced profile becomes profile 0002. If the profile update is for a subset of the active ports, the ports not being updated remain unchanged. Profile debug messages can be suppressed by coding DEBUG OFF or DEBUG SUMMARY in TELNETGLOBALS and placing it before all other Telnet statement blocks. The structural layout of the profiles and how connections are associated with profiles are shown in the following figure.

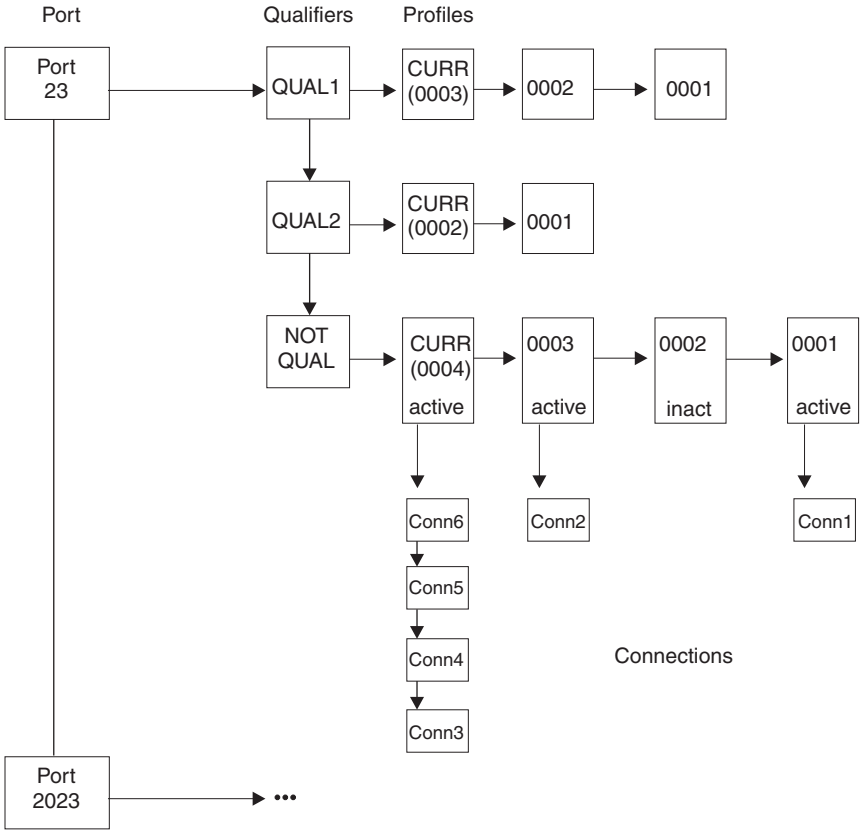


Figure 51. Telnet profiles and connections

Connection association

New connections are associated with the current profile and use the mappings and parameters defined by that profile. Even if a VARY TCPIP,,OBEYFILE command

updates the port, existing connections remain associated with the same profile. The statements of non-current profiles remain in effect and continue to support all connections that were established when the non-current profile was the CURRent profile. When all connections associated with a non-current profile have ended, the storage for the non-current profile mapping rules is freed and the profile is considered INACTIVE.

Connection mode choices

The TN3270 Telnet server supports several connection types. The negotiation process is hierarchical in the order listed below:

- TN3270 Enhanced (TN3270E)
- TN3270
- Linemode
  - Standard
  - Binary
  - Transform

TN3270E is the default connection mode for the TN3270 Telnet server. If the client refuses TN3270E mode, the server tries TN3270 mode. If the client refuses TN3270 mode, the server then tries Linemode. The TN3270 Telnet Server does not support Network Virtual Terminal (NVT) mode, except to allow the negotiation of TN3270E, TN3270, or linemode connections.

Note: The Type of Service (ToS) byte, also known as the Differentiated Services field, is not managed directly by the Telnet server. If you want to use Differentiated Services for Telnet, use the Quality of Service (QoS) support discussed in Chapter 12, “Quality of Service (QoS)” on page 565.

TN3270E and TN3270 are very similar. If the TN3270E functions described in the following sections are not needed, the end user does not notice any difference between TN3270E and TN3270 connections. In some cases, older clients do not properly refuse the server request for a TN3270E connection, and the connection is dropped. In these unusual cases, use the NOTN3270E parameter to disable the TN3270E function for those clients. Similarly, use the NOSNAEXT parameter for any client that does not properly negotiate the extension functions (Contention Resolution and SNA Sense). TN3270E/NOTN3270E and SNAEXT/NOSNAEXT parameters can be coded at all three parameter block levels for different levels of granularity.

TN3270E and TN3270 clients can receive a Telnet solicitor panel to submit an application name, User ID, and password to the server. The cursor is positioned on the application line unless the OLDSOLICITOR parameter is specified which causes the cursor to be positioned on the user line. Refer to “Using the Telnet Solicitor or USS logon panel” on page 362 for detailed information.

The ATTN key function is supported over TN3270, TN3270E, and Transform Linemode connections. It is not supported over Standard or Binary Linemode. Default LOGMODEs for TN3270E connections are SNA, and default LOGMODEs for TN3270 and Transform connections are non-SNA. Telnet processes the ATTN key differently for SNA and non-SNA LOGMODEs. In addition, the Telnet server can be configured to handle double ATTNs sent by some clients by specifying SINGLEATTN. See “Device types and logmode considerations” on page 361 for more information.

For TN3270E, LU assignment is done during connection negotiation. For TN3270, LU assignment is done at application selection time. To delay LU assignment until application selection time for TN3270E, specify the SIMCLIENTLU parameter. Refer to “LU mapping by application name” on page 349 and “LU mapping selection rules” on page 352 for details.

You might experience unexpected results if you start a Telnet session from within an application that is already connected using Telnet. For example, if you start a new Telnet session from within a TSO session that was established on a TN3270E connection, the keyboard will unlock when it seems it should not. This happens when an unlock keyboard intended for only the original, first session is sent from Telnet. The second session should remain locked but does not. An unlock keyboard intended for only the first session has the affect of unlocking the keyboard for both the first and second session since both are represented by the same client.

TN3270 Enhanced (TN3270E)

TN3270E connections support full-screen 3270 emulation that is sometimes referred to as TN3270 Extended. Do not confuse TN3270E function with the IBM 327x device types that end in -E (for example, 3278-2-E). In these cases, the *E* indicates that the terminal supports Extended field attributes such as color and highlighting and is not related to Telnet functions.

Telnet is often used as the primary method of connection between client workstations and the SNA mainframe environment. To make this form of remote connection as seamless as possible, Telnet terminal emulation simulates actual SNA terminals as closely as possible. To accomplish this, RFC1647 and RFC2355 (both known as TN3270E) add the ability to specify device names at connection time, add support for printer devices, and add additional SNA functions. An Internet draft, RFC 2355 Extensions, adds Contention Resolution and SNA Sense code support.

Device name specification

The Telnet server assigns LUs based on the LU mapping statements supplied. Clients are assigned a device name (Telnet LU name) based on those statements. However, a TN3270E client can optionally specify that a particular device name be assigned, or it can specify that a device name from a pool of LUs be assigned. If the specified device name is allowed for this client based on the LU mapping statements and the LU is available, the server assigns the specified device name. If the specified device pool is allowed for this client based on the LU mapping statements and an LU within the pool is available, the server assigns a device name from the specified pool. Otherwise, the request is rejected with an appropriate reason code, and the connection is dropped. See “Mapping Objects to Client Identifiers” on page 325 for additional LU mapping information.

328x printer support

Many Telnet clients emulate 328x class printers (device type IBM-3287-1). Most support both SNA Character Stream (SCS) as an LU1 and 3270 data stream as an LU3. The support of each is negotiated at connection time. When connected in TN3270E mode, the Telnet server supports these emulators in a manner similar to terminal LUs. Telnet can be configured to initiate a session at connection time or simply open an ACB to let the application initiate the session. The bind initiating each session is sent to the client, and the bind informs the emulator which data stream to expect. The VTAM application perceives the Telnet LU to be an an actual 3287-class printer and sends the SCS or 3270 data to the Telnet LU. Telnet

passes the data on to the client, which prints the data. Telnet printer support allows you to use a single product, Telnet, to control both SNA terminals and SNA printers.

Some Telnet client printer emulators can request to be associated with a terminal device name by specifying the terminal device name during connection negotiation. Using printer association, end users can connect their Telnet terminals to an application and then have Telnet assign an associated printer device name based on the terminal name. To associate printers with terminals, Telnet must have a printer device pool of LUs defined and a terminal device pool of LUs defined with each having the same number of device names.

Additional negotiated 3270 support

Responses and SysReq functions are supported by most clients that support TN3270E connections. Contention Resolution and SNA Sense support are newer and less prevalent.

- Responses - The client or host VTAM application can request that it receive and provide definite, exception, or no response. Client responses to application requests provide more accurate response information. For TN3270 connections, the server must intercept response requests from the host and respond on behalf of the client, incorrectly reducing measured response time.
- SysReq function - The end user can request that a current session be dropped by entering LOGOFF (in upper, lower, or mixed case) after pressing the SysReq key. If LUSESSIONPEND is not mapped to the client, the connection will be dropped. Otherwise, a USSMSG10 screen is sent to the client. If, instead of entering LOGOFF, the SysReq key is pressed a second time and if the application supports LUSTAT 082B (presentation screen is lost), the previous screen is resent to the client emulator.
- Contention Resolution - Improves communication between the client and host VTAM application regarding which owns the send state. Contention Resolution includes the following:
  - Start Data Indicator (SDI) - When the host sends change direction or end bracket, the server sends the SDI to the client. This allows the client to know exactly when data can be sent to the server.
  - BID - A BID sent from the host VTAM application is forwarded to the client instead of being intercepted and handled by the server. This allows the client to manage the BID process for itself.
  - Signal Indicator - A signal received from the host VTAM application is forwarded to the client. When the client responds to the signal, the server sends a change direction indicator to the host VTAM application.
- SNA Sense Support - Allows the client to include SNA sense codes in a response message. The client retains the option of letting the server map the errors to an appropriate sense code by not turning on the SNA-Sense-Code indicator in the response message.

TN3270

TN3270 connections support full-screen 3270 emulation. TN3270 connections do not support:

- Device name or pool name specification
- Printers

- Client involvement with responses, SysReq, Start Data Indicator, BID, Signal or SNA Sense data.

RFC1646 defines device name specification and printer support for TN3270 connections. However, this RFC is not supported on the TN3270 Telnet server. If either of these requests is received on a TN3270 connection, the server will drop the connection.

Linemode

In some cases, the client or the application does not support full-screen presentation, or the end user needs to work in a linemode environment. For these reasons, most emulators support linemode. Linemode supports a *go-ahead* function to simulate a half-duplex format. With *go-ahead* negotiated, the partner cannot send data until it receives a *go-ahead* from the current sender of data. In most cases, sessions are naturally half-duplex and the *go-ahead* adds unneeded transmissions. Therefore, the Telnet default is to Suppress Go Ahead (SGA). If *go-ahead* is needed to maintain a half-duplex format, use the NOSGA parameter. SGA or NOSGA can be coded at all three parameter block levels for different levels of granularity.

Telnet supports the following types of Linemode connections:

- Standard
- Binary
- Transform

Standard Linemode is assumed if neither DBCS transform nor BINARY linemode parameters are specified, or if the device type is not supported by transform. Standard Linemode is the only connection mode that requires translation by Telnet. Telnet supports NLS for standard Linemode connections. ASCII and EBCDIC code pages are the basis for translation. Telnet makes use of the National Language Support ICONV services available in the C runtime library. For custom code page information, refer to the ICONV services in *z/OS C/C++ Programming Guide*. When ASCII and EBCDIC code pages are specified, a conversion descriptor will be given to Telnet. Telnet creates ASCII-EBCDIC and EBCDIC-ASCII translation tables based on the conversion descriptor. The CODEPAGE parameter is used to specify the code page names. For example:

```
CODEPAGE ISO8859-1 IBM1047
```

The possible results from CODEPAGE processing are:

- If a conversion descriptor is not returned, CODEPAGE is not coded, or there is an error in the syntax, a default code page of ISO8859-1 will be used for ASCII, and the language environment code page taken from locale information will be used as the EBCDIC code page.
- If a conversion descriptor is not returned again, a default code page of IBM-1047 will be used for EBCDIC.
- If a conversion descriptor is not returned again, predefined translation tables within Telnet will be used. These tables are similar, but not exactly the same as the tables which would have been generated if ISO8859-1 and IBM-1047 had worked. Some of the differences are noted below:

```
EBCDIC           ASCII
x'0D25' -----> x'0D0085' using ISO8859-1/IBM-1047
x'0D25' -----> x'0D0A'   using internal tables
x'15'   <----  x'0A'     using ISO8859-1/IBM-1047
x'25'   <----  x'0A'     using internal tables
```

No message is issued to the console if the first conversion succeeds. If there is any conversion failure a message is issued. If one of the later conversions succeeds, a message is issued indicating success.

If your Linemode connection does not perform correctly, the default translation tables may be causing the problem. Try the internal Telnet translation tables by specifying TNSTD for both ASCII and EBCDIC choices. For example:

```
CodePage TNSTD TNSTD
```

The internal code pages must be used together. If only one of the two internal tables is specified, then the other internal table will also be used.

CODEPAGE can be coded at all three parameter block levels for different levels of granularity.

Binary Linemode is set using the BINARYLINEMODE parameter in TELNETPARMS. It indicates that Telnet should not do translation. The ASCII data from the client should be passed as-is to the VTAM application. BINARYLINEMODE or NOBINARYLINEMODE can be coded at all three parameter block levels for different levels of granularity.

Transform Linemode is set using the DBCSTRANSFORM parameter. When coded, all data that passes through Telnet will be *transformed* from DBCS or SBCS ASCII full screen to 3270 full screen for all supported device types. If the device type is not supported, Standard or Binary Linemode is used. DBCSTRANSFORM can be coded in TELNETPARMS or PARMSGROUP for different levels of granularity. It cannot be coded in TELNETGLOBALS. A unique logmode for transform can be set using TELNETDEVICE with a device type of TRANSFORM. Any logmode used must not support extended graphics.

Note: Transform can be used by only one port when multiple ports are active on one TCP/IP stack. DBCSTRANSFORM supports a maximum of 250 concurrent connections.

DBCSTRANSFORM can be used for either the VT100 single-byte character set (SBCS) or VT282 double-byte character set (DBCS) transform mode. When DBCSTRANSFORM is specified and the TCP/IP procedure JCL has been modified as shown below, ASCII-based terminal emulators (VT100 or VT282) will appear as full-screen 3270 terminals. The Telnet server receives ASCII data from the client and transforms it into SBCS or DBCS EBCDIC data, depending on the terminal type. Telnet adds appropriate SNA control bytes to give the appearance that the data is coming from a 3270 terminal. The Telnet server receives EBCDIC data from the host application and transforms the SNA control bytes and data into appropriate ASCII control bytes and data. The data is sent to the ASCII-based terminal where it is displayed in 3270 full-screen emulation. DBCSTRANSFORM requires additional special Data Definition (DD) statements in the TCP/IP procedure.

You must add the following three DD statements to the TCP/IP procedure JCL to support Transform:

```
//TNDBCSCN DD DSN=hlq.SEZAINST(TNDBCSCN),DISP=SHR
//TNDBCSXL DD DSN=hlq.SEZAXLD2,DISP=SHR
//TNDBCSER DD SYSOUT=*
```

- The TNDBCSCN DD statement must point to the configuration data set for 3270 DBCS transform mode. This configuration data set specifies the default DBCS conversion mode that will take effect at initialization time. Specify the CODEKIND and CHARMODE parameters according to the required DBCS code page. If

CODEKIND and CHARMODE are not specified, or if the TNDBCSCN DD statement is not added, CODEKIND defaults to SJISKANJI and CHARMODE defaults to ALPHABET. A sample can be found in hlq.SEZAINST(TNDBCSCN).

- The TNDBCSXL DD statement must point to the data set containing binary translation table code files for 3270 DBCS transform mode. The installation data set, *hlq*.SEZAXLD2, contains the default binary translation table code files. The binary translation table code files for 3270 Transform can be customized by using the CONVXLAT command. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about customizing translation table code files. If the TNDBCSXL DD statement is not added, an abend will occur.
- The TNDBCSER DD statement defines where Transform-specific error messages are recorded. This DD statement can specify an output data set or SYSOUT=*. If the TNDBCSER DD statement is not added, transform initialization will fail.

Specifying the DBCSTRACE parameter sends detailed trace output from 3270 Transform to the location specified in the SYSPRINT output DD statement. Additional detailed trace output is also sent to TNDBCSER. Both data sets will contain detailed trace data. DBCSTRACE or NODBCSTRACE can be coded in TELNETPARMS or PARMSGROUP for different levels of granularity. They cannot be coded in TELNETGLOBALS.

Connection security

This section describes data overrun security and transport layer security (TLS).

Data overrun security

MAXRECEIVE: This parameter in TELNETPARMS limits the number of bytes received from a client without an End Of Record (EOR) being received. If the data received exceeds the limit, the connection is dropped. This parameter protects against a client stuck in a send-data loop. In general, large file transfers will not be affected because the sending client typically divides the file into smaller records that are sent. The receiving application rebuilds the file as the smaller records are received.

MAXVTAMSENDQ: This parameter in TELNETPARMS limits the number of data segments (RPLs) queued to be sent to VTAM. If the queue size exceeds the limit, the connection is dropped. This parameter protects against using up large amounts of storage to hold data destined for a host application that is not receiving data.

MAXREQSESS: This parameter in TELNETPARMS limits the number of session requests received by Telnet in a 10–second period. For this parameter, a BIND received by Telnet defines a session request. If the number of BINDs received in a 10–second period exceeds the limit, an error is reported. This parameter protects against session logon loops that are possibly created by an automatic CLSDST-PASS to an inactive session. This parameter cannot protect against logon loops caused by an inactive default application and a client using auto-reconnect.

The MAXRECEIVE, MAXVTAMSENDQ, and MAXREQSESS parameters can be coded at all three parameter block levels for different levels of granularity.

Auto-reconnect loop: Without MSG07 coded a client connection error causes Telnet to drop the connection. The error may be an inactive DEFAULTAPPL or an LU assignment error. If the client has AUTO-RECONNECT specified, a continuous loop of retries occurs. The best protection against this is to code the MSG07 parameter which keeps the client from being disconnected. However, other

applications can be chosen from the error screen returned to the end user. To block end users from other applications, use the DEFONLY parameter.

Transport layer security

TN3270 transport layer security overview: The TN3270 server provides the ability to secure Telnet connections with the transport layer security (TLS) or secure sockets layer (SSL) protocol. References to RACF apply to any other SAF-compliant security products which contain the required support. In this chapter, a port that is configured to use the TLS/SSL protocol is referred to as a *secure port* or SECUREPORT. A connection that does not use the TLS/SSL protocol is referred to as a basic connection. The flows between Telnet and VTAM are unchanged.

The Internet Engineering Task Force (IETF) TLS-based Telnet Security Draft is supported. This Draft allows a TN3270 negotiation to determine if the client wants or supports TLS/SSL prior to beginning the handshake. The default action that the TN3270 server will take for a secure port is to first attempt a TLS/SSL handshake. If the client does not start the handshake within the time specified by SSLTIMEOUT, an attempt will be made to negotiateTLS/SSL as defined by the TLS-based Telnet Security Draft. If the client responds that a secure connection is desired, the handshake is started; if the client rejects TLS/SSL, the connection will be closed. This allows installations to support both types of secure clients without knowing which protocol the client is using. The default action can be changed by specifying the CONNTYPE parameter described later in this section.

Telnet server authentication and client authentication are described in Appendix B, “TLS/SSL security” on page 721. The Telnet server supports level 1, level 2 and level 3 client authentication. Client authentication is done with the CLIENTAUTH parameter. Level 2 and level 3 client authentication use RACF services to translate the client certificate to an associated user ID. That user ID is used as a client identifier.

The Telnet server supports limiting and ordering the encryption algorithms. Use the ENCRYPTION parameter to define the number and order of encryption algorithms. The ENCRYPTION parameter can be coded in TELNETGLOBALS, TELNETPARMS or PARMSGROUP, providing a high level of granularity.

Initialization occurs when the first secure port is activated and is not done again unless all secure ports are stopped (V TCPIP,,T,STOP,PORT=SECURE). Whether or not hardware encryption is used is based on its availability at the time of Telnet initialization with System SSL. If all secure ports are stopped, the check for cryptographic hardware presence and validity will be done again when the next TN3270 secure port is brought online. For more information on hardware encryption, see Appendix B, “TLS/SSL security” on page 721.

Configuring the TN3270 server to support TLS/SSL connections: To implement secure connections, TCP must have APF authorized access to the System SSL DLLs. The System SSL DLLs are located in hlq.SGSKLOAD by default. System SSL uses the C runtime library (SCEERUN) and the C/C++ IBM Open class library (SCLBDLL) which must also be accessible to TCP. To access these libraries, either add them to the linklist or specify them in the TCP procedure’s STEPLIB. If accessed via the linklist, the linklist must be authorized (LNKAUTH=LNKLST specified in the IEASYSxx parmlib member) or the libraries explicitly APF authorized. If accessed via a STEPLIB, the libraries must be APF authorized and DISP=SHR specified. The TCP/IP profile must also be updated. An

overview of the SSL related profile parameters follows. For a detailed description of the parameters, refer to *z/OS Communications Server: IP Configuration Reference*.

The two essential parameters that must be specified are:

- SECUREPORT – All TLS/SSL enabled TN3270 ports must be defined by specifying a TELNETPARMS block for each port. The SECUREPORT port designation statement in the TELNETPARMS block indicates the port is capable of handling SSL connections.
- KEYRING – As mentioned in the overview section, a server certificate is required for the server authentication process defined by the SSL protocol. This certificate is stored in a keyring. The keyring type and location is specified in the KEYRING statement. Only one keyring can be used by the TN3270 server.

The keyring can be defined in the TELNETGLOBALS or TELNETPARMS block. TELNETGLOBALS is the preferred definition method since it ensures that the same keyring has been defined for all SECUREPORTs. If specified in TELNETPARMS, the same keyring type and file must be specified for each SECUREPORT. The first keyring file name read is considered the correct keyring file name. The TELNETGLOBALS keyring is read first and then the TELNETPARMS keyrings are read in reverse order. Any keyring that does not match the first is rejected and the port update fails.

The following steps are required to enable TLS/SSL support for Telnet, with server authentication.

1. Generate the Telnet server private key and server certificate.
2. Configure Telnet to include one or more TLS/SSL enabled ports and specify the name of the keyring created in the step above in the TELNETGLOBALS block or the TELNETPARMS block. For example:
   - KEYRING HFS /usr/ssl/server.kdb (In this example 2 files, server.kdb and server.sth, were created using the gskkyman utility. The server's certificate is contained in the server.kdb file and designated as the default certificate.) The key database and the password stash file must reside in the same directory.
   - KEYRING MVS tcpip.mvs180.kdb (In this example 2 files, mvs180.kdb and mvs180.sth, were converted to MVS data sets from gskkyman files.)
   - KEYRING SAF serverkeyring (In this example, RACF is used to manage keys and certificates. The server certificate is connected to a keyring called SERVERKEYRING and designated as the default certificate.)
3. Restart TCP/IP or issue VARY TCPIP,,OBEYFILE with the updated configuration files.

Optional security parameters: Optional security parameters can only be specified for SECUREPORTs and can be specified in the TELNETGLOBALS, TELNETPARMS or PARMSGROUP blocks. The parameters specified in the PARMSGROUP block apply only to the clients mapped to the PARMSGROUP block by the PARMSMAP statement and override the parameters specified in the TELNETPARMS or TELNETGLOBALS block. The parameters specified in the TELNETPARMS block apply to any connection for that port if not overridden by a PARMSGROUP parameter. The parameters specified in the TELNETGLOBALS block apply to any connection for any port if not overridden by a TELNETPARMS or PARMSGROUP parameter.

The ENCRYPTION parameter is used to limit the encryption algorithms to only those included in parameter statement. If this parameter is not specified, any encryption algorithm supported by the installed level of System SSL is available for

use. See the *z/OS Communications Server: IP Configuration Reference* for the encryption algorithms that can be specified. The following are some reasons to use this parameter:

- The applications supported on this port require a high level of security and the installation wants all data encrypted using a particular encryption algorithm
- Certain connections are local and the installation does not require encryption for local clients. NULL encryption can be specified for this subset of connections.

The CONNTYPE parameter sets the level of security for connections. If CONNTYPE is not specified, a SECUREPORT defaults to CONNTYPE SECURE and a basic port defaults to CONNTYPE BASIC.

Valid CONNTYPE options are:

- SECURE – Indicates that the TLS/SSL handshake will be used to start the connection. If the client does not start the handshake within the time specified by SSLTIMEOUT, an attempt will be made to do a negotiated TLS/SSL handshake (as defined by the IETF TLS-based Telnet Security Draft); if the client rejects TLS/SSL, the connection will be closed.
- NEGTSECURE – Indicates the client supports the IETF TLS-based Telnet Security Draft. A TN3270 negotiation with the client first determines if the client is willing to enter into a secure connection. If the client agrees, a TLS/SSL handshake is started and secure protocols will be used for all subsequent communication. If the client rejects TLS/SSL, the connection will be closed.

  If you know that the TN3270 secure clients connecting into the port are using the protocol defined by the TLS-based Telnet Security Draft, you should consider using this option. With this option the TLS/SSL handshake is not attempted until after a positive response to the TN3270 DO STARTTLS IAC is received. This avoids the timeout delay that can occur when a TLS/SSL handshake is immediately started (as done with CONNTYPE SECURE) but the client is expecting the protocol used by the TLS-based Telnet Security Draft.
- BASIC – Indicates that a basic (non-secure) connection will be used.
- ANY – Indicates that the connection can be either secure or basic. The TN3270 server will first try a standard TLS/SSL handshake. If the handshake times out, a negotiated TLS/SSL (see CONNTYPE NEGTSECURE) is attempted.
  - If the client is willing to enter into a secure connection, secure protocols will be used for all subsequent communication.
  - If the client is not willing to enter into a secure connection, a basic (non-secure) connection is used.
- NONE – Indicates that no connection is allowed and the connection will be closed. If this option is specified in TELNETPARMS, a PARMSMAP must cover every allowable connection and the related PARMSGROUP must specify the desired CONNTYPE.

The CLIENTAUTH parameter indicates that the client must send a client certificate to the server. If this parameter is not specified, a client certificate is not requested during the SSL handshake and no certificate based client authentication is done. The level of validation done depends on the option specified.

Valid CLIENTAUTH options are:

- SSLCERT (Level 1) – To pass authentication, the Certificate Authority (CA) that signed the client certificate must be considered trusted by the server (that is, a certificate for the CA that issued the client certificate is listed as trusted in the server's keyring).

- SAFCERT (Level 2 and 3) – The level 1 checking provided by SSLCERT is done and level 2 checking is done to verify that the certificate has been registered with RACF (or other SAF compliant security product that supports certificate registration). Additionally, if the SERVAUTH RACF class is active and a RACF resource has been defined for the port, level 3 client authentication is in effect and the connection is allowed only if the user ID associated with the client certificate has READ access to the RACF resource.
- NONE – No client certificate is requested.

The CRLLDAPSERVER parameter is specified in the TELNETGLOBALS block. It defines the name or IP address and port of the Certificate Revocation List (CRL) LDAP server. The CRL LDAP server is used only if client certificates are received (CLIENTAUTH specified). If CLIENTAUTH and the CRLLDAPSERVER have been specified, the certificate revocation list is checked during client authentication. If the client's certificate is found on the certificate revocation list, the connection is closed. Only one CRL LDAP server can be defined to the Telnet server.

Changing the Keyring (name, type or contents) or the CRL LDAP server (name or location) cannot be done using VARY TCPIP,,OBEYFILE while secure ports are active. To change the Keyring or the CRL LDAP server, all secure ports must first be stopped (V TCPIP,tcpname,T,STOP,PORT=S). VARY TCPIP,,OBEYFILE can then be used to restart the secure ports with a new keyring or CRL LDAP server. If the CRL LDAP server is stopped or connectivity is lost, System SSL may not recognize a subsequent reconnection. This situation must be handled like the CRL LDAP server change.

Using one port for both basic and SSL connections*:* SECUREPORT indicates that a port is capable of supporting SSL connections. The default connection type for SECUREPORT is SECURE. CONNTYPE can be used to modify connection types on a single port. Allowing a port to support both basic and secure connections assumes that either:

- The installation will allow the client to determine the connection type desired.
- A subset of the connections that should use a particular connection security type can be identified by Client Identifier.

In the first case, CONNTYPE ANY can be specified. If the port was defined as a SECUREPORT but the client wants a basic connection, there will be a slight delay before connection negotiation begins. This is because when CONNTYPE ANY is coded, the Telnet server will first attempt an SSL handshake to ensure the client is not requesting SSL support. It is only after the SSL handshake times out and negotiated security is rejected that the basic connection negotiation begins.

In the second case, the TELNETPARMS block should specify the default connection security type (see the CONNTYPE parameter). For connections with different connection security requirements:

- Identify the clients by Client Identifier.
- Create a PARMSGROUP with the alternate CONNTYPE definitions.
- Map the PARMSGROUP to the clients using the PARMSMAP statement.

Telnet profile example*:* The following example defines three ports with the characteristics discussed below.

- Port 23 allows only basic (non-secure) connections.
- Ports 992 and 1023 are enabled for secure connections and use the keyring defined in the TELNETGLOBALS block.

- Port 992 allows only secure connections. No client authentication is requested.
- Port 1023 allows both basic and secure connections. The installation desires the following characteristics for port 1023:
  - The system administrator is at IP address 9.37.88.1 and wants the capability to choose to connect with secure or non-secure connections.
  - Building A and B are local and do not need connection security. The clients in these buildings have identifiable host names. The installation wants only these clients to use basic connections to avoid the encryption overhead.
  - Connection security is desired on all other connections.
  - All secure connections require client authentication and will use the DES or triple DES encryption algorithms.

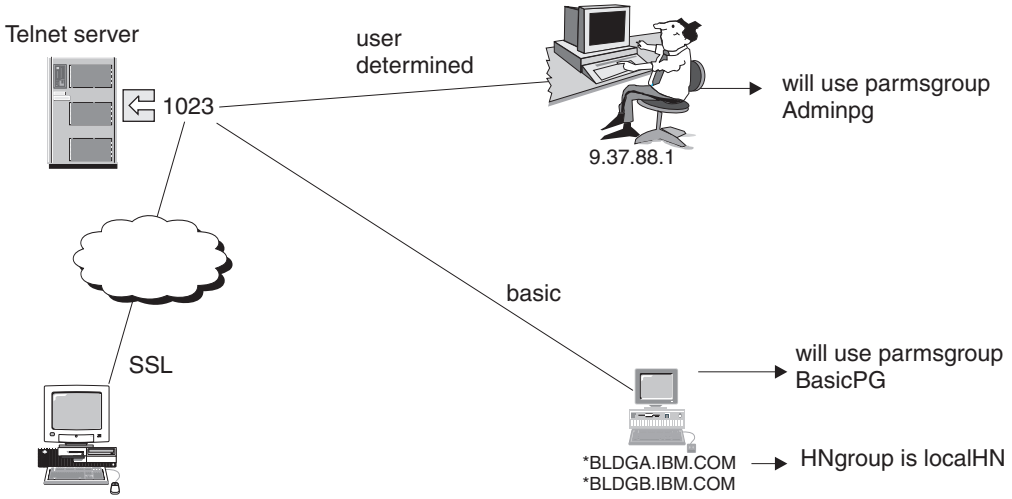


Figure 52. Port 1023 connection characteristics

Note: Only the definitions applicable to TLS/SSL connection security are shown; additional parameters might be needed.

```
TELNETGLOBALS
KEYRING hfs /usr/keyring/tcpcs.kdb ;keyring used by all SECUREPORTs
ENDTELNETGLOBALS


TELNETPARMS            ; basic port does not support secure connections
Port 23
ENDTELNETPARMS

TELNETPARMS            ; port that allows only secure connections
SECUREPORT 992         ; no client authentication requested
ENDTELNETPARMS         ; any supported encryption algorithm

TELNETPARMS            ; port that allows secure and BASIC connections.
SECUREPORT 1023        ; note: BEGINVTAM block has PARMSGROUP that may override CONNTYPE
CONNTYPE SECURE        ; SECURE is default
CLIENTAUTH SSLCERT     ; client certificate must be issued by a trusted CA
ENCRYPT SSL_DES_SHA SSL_3DES_SHA ENDENCRYPT ; only encrypt with DES or
ENDTELNETPARMS                              ;triple DES



BEGINVTAM
Port 1023
...                    ;Mapping statements
HNGROUP localHN
   *.BLDGA.IBM.COM
   *.BLDGB.IBM.COM
ENDHNGROUP
```

```
PARMSGROUP BasicPG     ; override telnetparms definitions
   CONNTYPE BASIC      ; support non-secure connections mapped to this group
ENDPARMSGROUP
PARMSGROUP AdminPG
   CONNTYPE ANY        ; connections mapped to this group allow any type of connection
ENDPARMSGROUP

PARMSMAP AdminPG 9.37.88.1 ; this ip address can use secure or non-secure connections
PARMSMAP BasicPG localHN   ; hosts defined in HNGROUP localHN,
                           ;will use non -SSL connections as defined in PARMSGROUP BasicPG
ENDVTAM


BEGINVTAM
Port 992 23
...                        ;Mapping statements
                           ;no PARMSGROUP defined for these ports
                           ;TELNETPARMS definitions used for all connections
ENDVTAM
```

Mapping Objects to Client Identifiers

The TN3270 Telnet server provides flexibility for mapping Objects to clients based on Client Identifiers. This section provides definitions, rules, and examples of many mapping methods. Examples start with simple concepts, then progress to more complicated concepts showing interaction between mapping statements. All mapping statements are specified in the BEGINVTAM block. Refer to *z/OS Communications Server: IP Configuration Reference* for statement rules not discussed here.

The general relationship of mapping statements is:

MAP OBJECTS to clients based on **CLIENT IDENTIFIER**

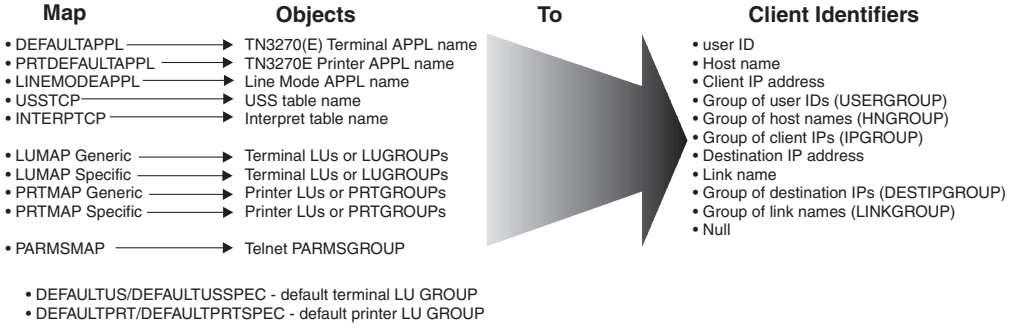


Figure 53. Mapping model

Telnet tries to assign all ten Objects to a client based on the mapping statements when the connection is accepted. The search for Objects continues until all Objects are found or until all mapping statements are checked.

Objects

When a client connection request is made, Telnet must assign an LU name to represent the client. Optionally, a USS table, default application, or unique parameters defined in the PARMSGROUP statement can be assigned to the connection. See “Mapping methods” on page 335 for details about how these objects are mapped to clients. The complete list of object follows:

- TN3270(E) terminal application name – The DEFAULTAPPL mapping statement maps the TN3270(E) terminal application Object to a terminal client. When a

TN3270 or TN3270E connection is negotiated, Telnet immediately initiates a session request to the VTAM application.

- TN3270E printer application name – The PRTDEFAULTAPPL mapping statement maps the TN3270E printer application Object to a printer client. When a TN3270E printer connection is negotiated, Telnet immediately initiates a session request to the VTAM application.
- Line Mode application name – The LINEMODEAPPL mapping statement maps the linemode application Object to a client. When a linemode connection is negotiated, Telnet will immediately initiate a session request to the VTAM application.
- USS table name – The USSTCP mapping statement maps the USS table Object to a client. When a TN3270 or a TN3270E connection is negotiated, Telnet will send a USSMSG10 screen to the client. A special case condition exists when an application name and a USS table are both mapped to the client by the exact same Client Identifier. In this case, Telnet will immediately initiate a session request to the VTAM application and use the USS table for error messages.
- Interpret table name – The INTERPTCP mapping statement maps the Interpret table Object to a client. When a TN3270 or a TN3270E connection is negotiated, Telnet uses the Interpret table to modify USS commands. The client must have a USS table mapped to it for the Interpret table to be used.
- Terminal LUs or LUGROUPs (Generic) – The Generic LUMAP mapping statement maps a single LU or LUGROUP Object to a client. For single LU mappings, Telnet will assign the LU name to the connection if the LU is available. For LUGROUP mappings, Telnet will assign an available LU from the group to the connection. An LU is required to represent the client when initiating a VTAM session. DEFAULTLUS is a default terminal LUGROUP Object mapped generically to the NULL Client Identifier.
- Terminal LUs or LUGROUPs (Specific) – The Specific LUMAP mapping statement maps a single LU or LUGROUP Object to a client. Unlike the Generic mapping in which Telnet assigns the LU, the Specific mapping requires the client to specify the LU name it wants. Telnet verifies the LU is mapped and available. The specified LU name can be either a mapped single LU, an LU within a mapped LUGROUP, or the LUGROUP name of the mapped LUGROUP. If the client specifies an LUGROUP name, Telnet assigns an available LU from within the group. DEFAULTLUSSPEC is a default terminal LUGROUP Object mapped specifically to the NULL Client Identifier.
- Printer LUs or PRTGROUPs (Generic) – The Generic PRTMAP mapping statement maps a single LU or PRTGROUP Object to a client. For single LU mappings, Telnet will assign the LU name to the connection if the LU is available. For PRTGROUP mappings, Telnet will assign an available LU from the group to the connection. An LU is required to represent the client when initiating a VTAM session. DEFAULTPRT is a default printer PRTGROUP Object mapped generically to the NULL Client Identifier.
- Printer LUs or PRTGROUPs (Specific) – The Specific PRTMAP mapping statement maps a single LU or PRTGROUP Object to a client. Unlike the Generic mapping in which Telnet assigns the LU, the Specific mapping requires the client to specify the LU name it wants. Telnet verifies the LU is mapped and available. The specified LU name can be either a mapped single LU, an LU within a mapped PRTGROUP, or the PRTGROUP name of the mapped PRTGROUP. If the client specifies a PRTGROUP name, Telnet assigns an available LU from within the group. DEFAULTPRTSPEC is a default printer PRTGROUP Object mapped specifically to the NULL Client Identifier.

- Telnet PARMSGROUP – The PARMSMAP mapping statement maps the PARMSGROUP Object to a client. The parameters in the group override parameter values specified in either TELNETGLOBALS or TELNETPARMS.
- ALLOWAPPL – This statement allows client access to applications and optionally maps or confirms the mapping of an LU name to the client based on the application name chosen. DEFAULTAPPL application names are presumed allowed and do not require the ALLOWAPPL statement for Telnet acceptance. However, ALLOWAPPL may be used by default applications for LU assignment and other advanced functions.
- RESTRICTAPPL – This statement restricts Telnet acceptance of application names to only users that specify an acceptable User ID and password. It also optionally maps or confirms the mapping of an LU name to the client based on the application name and User ID chosen.

Client Identifiers

One client can be represented by many different Client Identifiers. For example, Telnet might assign an LU based on client host name, assign an application based on a client IP address, and assign a USS table based on connection link name. Refer to “Mapping methods” on page 335 for details about how these Client Identifiers are used to map Objects. In some cases, two different Client Identifiers that represent the same client are used on mapping statements to map the same type of Object. In these cases, Telnet must determine which Client Identifier to use when assigning the Object. See “Client Identifier selection rules” on page 329 for more details. The complete list of Client Identifiers and mapping examples follow:

- User ID or USERGROUP name - If the CLIENTAUTH SAFCERT parameter is used with a secure connection, the client is required to send its client certificate to the Telnet server for client authentication. The SAFCERT option indicates that the client certificate can be translated to a User ID by a security product such as RACF. Telnet translates the certificate as soon as the SSL handshake is done. The resulting User ID is associated with the connection. Objects can be mapped to the connection based on an exact User ID, or Objects can be mapped to a USERGROUP name containing exact User IDs and wildcarded User IDs. For example, mobile employees need to be assigned a unique set of LU names and the manager must always be assigned LU name LUMOBL01. These employees are not within a secure network and always use client authenticated secure connections. Their certificates are translated to User IDs by Telnet. Note the required use of the Client Identifier type USERID on the mapping statement. If it were not used, Telnet would assume the name is a linkname.

  ```
  USERGROUP   USGMOBL1
     MOBL0002 MOBL0003
     MOBL1%%C
  ENDUSERGROUP
  LUGROUP LUGMOBL1
     LUMOBL02..LUMOBL20
  ENDLUGROUP
  LUMAP  LUMOBL01  USERID,MOBL0001   ; mgr mapping
  LUMAP  LUGMOBL1  USGMOBL1          ; employee mapping
  ```

- Host name or HNGROUP name - If the network dynamically assigns IP addresses, the same client will not have the same IP address from one connection to the next. However, if Dynamic Domain Name System (DDNS) and Dynamic Host Configuration Protocol (DHCP) are used, the client host name can be constant. See Chapter 10, “Domain Name System (DNS)” on page 417 for more DDNS and DHCP information. With static host names, Objects can be mapped to clients based on their host name, or Objects can be mapped to HNGROUP names containing exact host names and wildcarded host names. For

example, LUADMNM is mapped to exact host name ADMIN.DEPT1.GROUP1.COM, and application INVENTRY is mapped to HNGROUP name HNGINV.

```
HNGROUP HNGINV
   INV1.DEPT1.GROUP1.COM
   *.DEPT3.GROUP1.COM
   **.GROUP3.COM
ENDHNGROUP
LUMAP LUADMNM  ADMIN.DEPT1.GROUP1.COM
DEFAULTAPPL    INVENTRY  HNGINV
```

Host name specification requires that Telnet resolve a host name from an IP address by using the resolver. To do this, a valid TCPIP.DATA data set must be provided. See Chapter 1, “Configuration overview” on page 3 for a description of how TCPIP.DATA is located. Telnet uses the native MVS sockets search order to find a resolver. Neither the Resolver_Config nor the /etc/resolv.conf HFS will be used when searching for TCPIP.DATA. The most common reason for message EZZ6011I, INIT_API failure, is that no resolver has been defined.

- Client (source) IP address or IPGROUP name - Client IP address is the most common method used to map Objects to the client. In a static network, Objects can be mapped to clients based on the exact IP address, or Objects can be mapped to IPGROUP names containing exact IP addresses and subnets. For example, LUADMN is mapped to exact IP address 1.1.1.1, and application PAYROLL is mapped to IPGROUP name IPGPAY.

  ```
  IPGROUP IPGPAY
     1.1.2.2  1.1.2.3
     255.255.0.0:2.2.0.0
  ENDIPGROUP
  LUMAP LUADMN 1.1.1.1
  DEFAULTAPPL  PAYROLL IPGPAY
  ```

  The IP/subnet combination of 0.0.0.0:0.0.0.0 is a special case that includes all connections. This might be useful if you want to have a default mapping with a higher priority than the NULL client identifier.

- Destination IP address or DESTIPGROUP name - A destination IP address is the host address that is the destination for a Telnet connection. Linkname can be used as a Client Identifier to map Objects to destination IP addresses when the linkname is static and defined in the profile. However, if the destination IP address is a dynamic Virtual IP Address (VIPA) , the linkname is not known before the VIPA is created. In this case, destination IP address is the ideal solution. In other cases, specifying the destination IP address in the Telnet profile may be more clear than specifying the linkname. For example, two TCP/IP stacks are backups for each other. Telnet connections to stack 1 (VIPA 5.5.5.1) default to logon manager application APPL1 and connections to stack 2 (VIPA 5.5.5.2) default to logon manager application APPL2. If one of the stacks becomes unavailable, the other will take over and dynamically add the failing stack’s VIPA. The dynamic linkname created is not easily predicted. Use the following statements in the profile of each stack to ensure users connecting to 5.5.5.1 always get APPL1 and users connecting to 5.5.5.2 always get APPL2 regardless of which stack is used. Note the required use of the Client Identifier type DESTIP on the mapping statement. If it were not used, Telnet would assume the IP addresses are client (source) IP addresses.

  ```
  DEFAULTAPPL  APPL1 DESTIP,5.5.5.1
  DEFAULTAPPL  APPL2 DESTIP,5.5.5.2
  ```

- Linkname or LINKGROUP name - A linkname is defined by the TCP/IP LINK statement. The linkname defines a host IP address that is a destination address

for clients connecting to Telnet. Linkname can be useful in cases where Object assignment is dependent on the client destination IP address instead of the client source IP address. Several linknames may be defined and the same LU mapping or other Object mapping may be desired for several linknames. In this case, a LINKGROUP can be defined and used on a single mapping statement. For example, based on the statements below, a client connecting to LINK1 IP address will be assigned an LU from the LUGROUP name LUGLNKS and will establish a session with TPX1. A client connecting to LINK2 IP address will be assigned an LU from the LUGROUP name LUGLNKS and will establish a session with TPX2. Because LINK1 and LINK2 are not group names, host names, or IP addresses, they are assumed to be linknames. The Client Identifier type, LINKNAME, can be used for clarity but is not required.

```
LINKGROUP    LNKGRP1
   LINK1 LINK2
ENDLINKGROUP
LUMAP        LUGLNKS  LNKGRP1
DEFAULTAPPL  TPX1     LINKNAME,LINK1
DEFAULTAPPL  TPX2     LINK2
```

When the destination IP address is the IP address of a dynamic XCF address, multiple linkname values can be associated with the IP address. Telnet will use the first linkname associated with the IP address in the home list. If a dynamic XCF destination is used as a Client Identifier, it is recommended that DESTIP be used instead of linkname. Results can vary using linkname.

- NULL (no Client Identifier) - The NULL Client Identifier type indicates that no Client Identifier was specified. The NULL Client Identifier is valid on the DEFAULTAPPL, LINEMODEAPPL, USSTCP, and INTERPTCP mapping statements. It is the implied Client Identifier for the DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTPRT, and DEFAULTPRTSPEC Objects. ParmsGroup is the only Object that cannot be mapped to the NULL Client Identifier. The NULL Client Identifier mapped Objects are the last Objects checked when assigning Objects to a client. For example, assume a client does not match any Client Identifier in the profile for DEFAULTAPPL or USSTCP. You can put the end user into session with a security application, named SecAppl, that can verify the end user is authorized to use the company's system. The Client Identifier field is blank.

  ```
  DEFAULTAPPL SECAPPL
  ```

Client Identifier selection rules

When Client Identifiers are used together, conflicts might occur. For example, host name NAME1.HOST1.COM may also be IP address 1.2.3.4. If the following DEFAULTAPPL statements exist, only one of the applications can be chosen.

```
DEFAULTAPPL  TSO  NAME1.HOST1.COM
DEFAULTAPPL  CICS 1.2.3.4
```

If USSTCP and DEFAULTAPPL have the same Client Identifier, DEFAULTAPPL will be used. For detailed information, refer to "Resolving DEFAULTAPPL and USS table conflicts" on page 340.

Telnet uses a very specific Client Identifier hierarchy when assigning Objects. The following order is used:

The mapping rule search order

- **Exact client identifier:**
  - 1) User ID, 2) hostname, 3) IP address
- **Exact client identifier in a group definition:**

– 4) User group, 5) hostname group, 6) IP address group

- **Wildcard match for client identifier in a group definition:**
  – 7) User group, 8) hostname group, 9) IP address group
- **Exact destination:**
  – 10) destination IP address, 11) link name
- **Exact destination in a group definition:**
  – 12) destination IP address group, 13) link name group
- **Wild card match for destination in a group definition:**
  – 14) destination IP address group, 15) link name group
- **Null client ID**
  – 16) DEFAULTAPPL, LINEMODEAPPL, USSTCP, INTERPTCP, DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTPRT, DEFAULTPRTSPEC

Examples:

- **Exact client identifier:**

```
1)  LUMAP   LU1     USERID,USER1
2)  LUMAP   LU2     NAME1.HOST1.COM
3)  LUMAP   LU3     1.2.3.4
```

  Client Identifier type USERID is required. If not specified, USER1 is assumed to be a link name.

- **Exact client identifier in a group definition:**

```
LUGROUP   LUGRP1    LU100..LU199   ENDLUGROUP
LUGROUP   LUGRP2    LU200..LU299   ENDLUGROUP
LUGROUP   LUGRP3    LU300..LU399   ENDLUGROUP

    USERGROUP USRGRP1
      USER1 USER2 USER3
    ENDUSERGROUP

    HNGROUP  HNGRP1
         NAME2.HOST1.COM  NAME2.HOST3.COM
    ENDHNGROUP

    IPGROUP  IPGRP1
      1.2.3.5     1.2.3.6
    ENDIPGROUP

4)  LUMAP   LUGRP1   USRGRP1
5)  LUMAP   LUGRP2   HNGRP1
6)  LUMAP   LUGRP3   IPGRP1
```

- **Wild card match for client identifier in a group definition:**

```
    USERGROUP USRGRP2
      USER%%  TCPU*
    ENDUSERGROUP

    HNGROUP  HNGRP2
         *.HOST2.COM  **.HOST3.COM
    ENDHNGROUP

    IPGROUP  IPGRP2
      255.255.0.0:2.3.0.0
    ENDIPGROUP

7)  LUMAP   LUGRP1   USRGRP2
8)  LUMAP   LUGRP2   HNGRP2
9)  LUMAP   LUGRP3   IPGRP2
```

- **Exact destination:**

```
10) DEFAULTAPPL    TSO        DESTIP,1.2.3.4
11) USSTCP         USSTAB1    LINK1
```

Client Identifier type DESTIP is required. If not specified, destination IP address 1.2.3.4 is assumed to be a client IP address.

- **Exact destination in a group definition:**

```
DESTIPGROUP DSTIPGRP1
  1.2.3.5  1.2.3.6
ENDDESTIPGROUP
LINKGROUP LINKGRP1
  LINK1 LINK2 LINK3
ENDLINKGROUP
12) LUMAP LUGRP1 DSTIPGRP1
13) LUMAP LUGRP2 LNKGRP1
```

- **Wild card match for destination in a group definition:**

```
DESTIPGROUP DSTIPGRP2
  255.255.0.0:1.4.0.0
ENDDESTIPGROUP
LINKGROUP LINKGRP2
  LINK* %LINK
ENDLINKGROUP
14) LUMAP LUGRP1 DSTIPGRP2
15) LUMAP LUGRP2 LNKGRP2
```

- **Null client ID**

```
16) DEFAULTAPPL     TSO
    LINEMODEAPPL    CICS
    USSTCP          USSTAB1
    INTERPTCP       INTTAB1
    DEFAULTLUS
      LU01..LU99
    ENDDEFAULTLUS
```

NULL is a single Client Identifier. The order of the examples has no significance. If DEFAULTAPPL and USSTCP mapping statements both have the NULL Client Identifier, the DEFAULTAPPL will be used regardless of order. For more information, refer to “Resolving DEFAULTAPPL and USS table conflicts” on page 340.

Object assignment examples

A client can be known by several different Client Identifiers. These Client Identifiers are used to assign as many Objects as possible to the connection based on the profile mapping statements. Telnet starts with the highest priority Client Identifier of the client and assigns all Objects mapped by that Client Identifier. If all 10 Objects are not assigned, Telnet uses the next highest priority Client Identifier (for prioritization details, see “Client Identifier selection rules” on page 329) and assigns all Objects mapped by that Client Identifier. This Object assignment process continues by using lower and lower priority Client Identifiers until all 10 Object types are found or until all of the matching Client Identifier mappings have been checked. If an Object is mapped by multiple Client Identifiers, only the Object mapped by the highest Client Identifier is used. It is unlikely all Objects are assigned to connections because not all Objects are always mapped. For example, many profiles do not contain PRTDEFAULTAPPL or INTERPTCP mapping statements. In this case, the printer default appl and Interpret table Objects will not be assigned.

Figure 54 on page 333 is a graphical representation of the following Telnet mapping statements. The numbered mapping statements correspond to the numbered buttons in the figure. The mappings that specify USERGROUP USGRP1 generate buttons 4 through 8 for exact user ID in a group and buttons 12 through 16 for wildcard user ID in a group.

```
LUGROUP    LUGRP1   LU01..LU10..FFNN    ENDLUGROUP
LUGROUP    LUGRP2   LU11..LU99..FFNN    ENDLUGROUP
PRTGROUP   PRTGRP1  PRT01..PRT10..FFFNN ENDPRTGROUP
PARMSGROUP PGDBG    DEBUG DETAIL        ENDPARMSGROUP
PARMSGROUP PGSCAN   SCANINTERVAL 10     ENDPARMSGROUP
PARMSGROUP PGMTKO   TKOSPECLU  7        ENDPARMSGROUP
PARMSGROUP PGALL    DEBUG DETAIL
                    SCANINTERVAL 10
                    TKOSPECLU  7        ENDPARMSGROUP

USERGROUP  USGRP1   PAYUSR1   PAYUSR*   ENDUSERGROUP
HNGROUP    HNGRP1   USER1.GROUP3.COM
                    USER5.GROUP3.COM    ENDHNGROUP

   (1) PARMSMAP       PGALL    USERID,PAYUSR1
   (2) LINEMODEAPPL   TSO      9.9.9.9
   (3) PARMSMAP       PGDBG    9.9.9.9

(4,12) DEFAULTAPPL    PAYROLL  USGRP1
(5,13) PRTDEFAULTAPPL PAYPRT   USGRP1
(6,14) LUMAP          LUGRP1   USGRP1   SPECIFIC
(7,15) PRTMAP         PRTPGRP1 USGRP1   SPECIFIC
(8,16) PARMSMAP       PGTKO    USGRP1

   (9) USSTCP         USSTABHN HNGRP1
  (10) LUMAP          LUGRP2   HNGRP1   GENERIC
  (11) PARMSMAP       PGSCAN   HNGRP1

  (17) INTERPTCP      INTTAB1  LINK1
  (18) DEFAULTAPPL    TPX1
  (19) USSTCP         USSTAB1
```

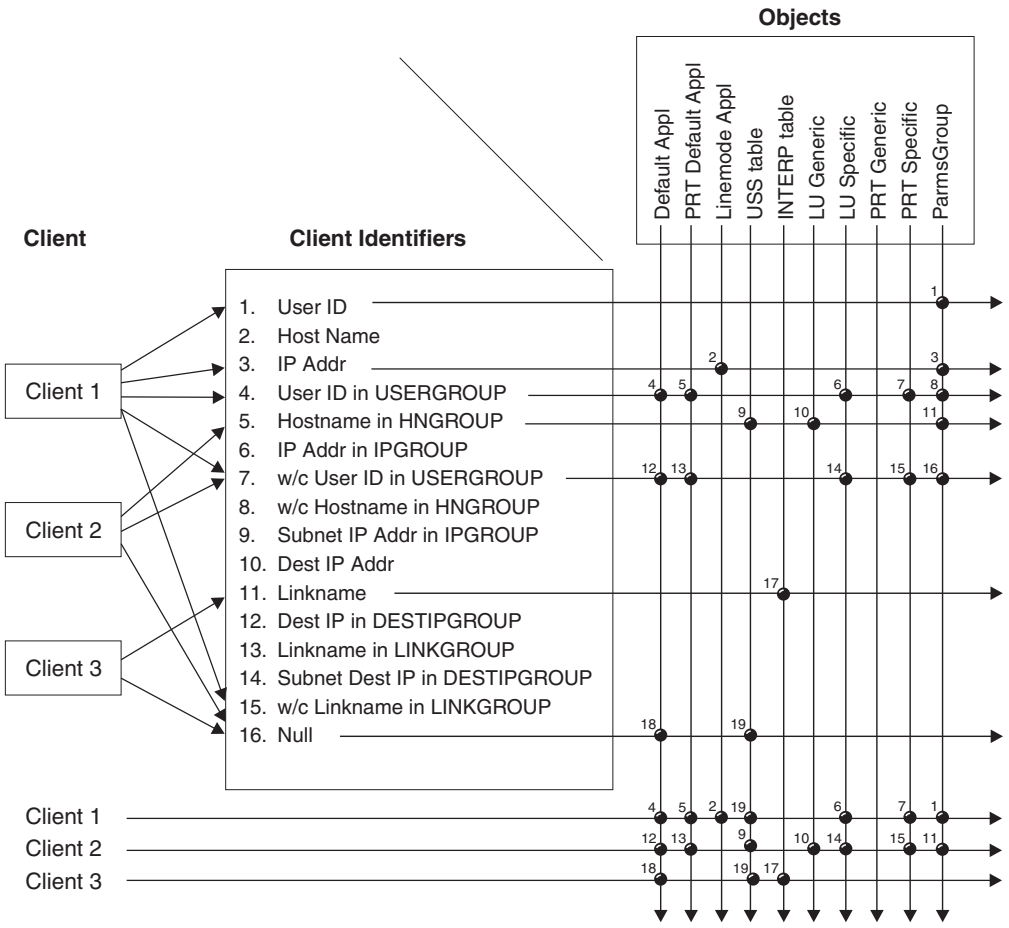


Figure 54. Search method

Client 1 example: Assume client 1 connects from IP address 9.9.9.9 using client authentication and is assigned PAYUSR1. The client does not have a host name ending in GROUP3.COM and does not have a linkname LINK1. Using Figure 54, the client will be assigned objects as shown in Table 15:

Table 15. Client 1 example

Button	Object type	Name	Action
(1)	ParmsGroup	PGALL	Assigned, exact user ID match
(2)	Linemode Appl	TSO	Assigned, exact IP address match
(3)	ParmsGroup		Not assigned, already assigned
(4)	TN3270(E) Appl	PAYROLL	Assigned, exact user ID match in group
(5)	Print Appl	PAYPRT	Assigned, exact user ID match in group
(6)	LUgroup-Spec	LUGRP1	Assigned, exact user ID match in group
(7)	PRTgroup-Spec	PRTGRP1	Assigned, exact user ID match in group
(8)	ParmsGroup		Not assigned, already assigned
(9)	USS table		Ignored, no exact hostname match in group
(10)	LUgroup-Gen		Ignored, no exact hostname match in group
(11)	ParmsGroup		Ignored, no exact hostname match in group

Table 15. Client 1 example (continued)

Button	Object type	Name	Action
(12)	TN3270 (E) Appl		Already covered, exact user ID in group
(13)	Print Appl		Already covered, exact user ID in group
(14)	LUgroup-Spec		Already covered, exact user ID in group
(15)	PRTgroup-Spec		Already covered, exact user ID in group
(16)	ParmsGroup		Not assigned, already assigned
(17)	Interp table		Ignored, no linkname match
(18)	TN3270(E) Appl		Not assigned, already assigned
(19)	USS table	USSTAB1	Assigned, NULL Client Identifier match

Client 2 example: Assume client 2 connects from IP address 9.1.1.1 using client authentication and is assigned PAYUSR5 and has a host name of USER5.GROUP3.COM. The client does not have a linkname LINK1. Using Figure 54 on page 333, the client will be assigned objects as shown in Table 16:

Table 16. Client 2 example

Button	Object type	Name	Action
(1)	ParmsGroup		Ignored, no exact user ID match
(2)	Linemode Appl		Ignored, no exact IP address match
(3)	ParmsGroup		Ignored, no exact IP address match
(4)	TN3270(E) Appl		Ignored, no exact user ID match in group
(5)	Print Appl		Ignored, no exact user ID match in group
(6)	LUgroup-Spec		Ignored, no exact user ID match in group
(7)	PRTgroup-Spec		Ignored, no exact user ID match in group
(8)	ParmsGroup		Ignored, no exact user ID match in group
(9)	USS table	USSTABHN	Assigned, exact hostname match in group
(10)	LUgroup-Gen	LUGRP2	Assigned, exact hostname match in group
(11)	ParmsGroup	PGSCAN	Assigned, exact hostname match in group
(12)	TN3270 (E) Appl	PAYROLL	Assigned, wildcard user ID match in group
(13)	Print Appl	PAYPRT	Assigned, wildcard user ID match in group
(14)	LUgroup-Spec	LUGRP1	Assigned, wildcard user ID match in group
(15)	PRTgroup-Spec	PRTGRP1	Assigned, wildcard user ID match in group
(16)	ParmsGroup		Ignored, no wildcard user ID match in group
(17)	Interp table		Ignored, no linkname match
(18)	TN3270(E) Appl		Not assigned, already assigned
(19)	USS table		Not assigned, already assigned

Client 3 example: Assume client 3 connects from IP address 9.2.2.2 without client authentication and has a host name of USER3.GROUP1.COM. The client connects to linkname LINK1. Using Figure 54 on page 333, the client will be assigned objects as shown in Table 17 on page 335:

Table 17. Client 3 example

Button	Object type	Name	Action
(1)	ParmsGroup		Ignored, no exact user ID match
(2)	Linemode Appl		Ignored, no exact IP address match
(3)	ParmsGroup		Ignored, no exact IP address match
(4)	TN3270(E) Appl		Ignored, no exact user ID match in group
(5)	Print Appl		Ignored, no exact user ID match in group
(6)	LUgroup-Spec		Ignored, no exact user ID match in group
(7)	PRTgroup-Spec		Ignored, no exact user ID match in group
(8)	ParmsGroup		Ignored, no exact user ID match in group
(9)	USS table		Ignored, no exact hostname match in group
(10)	LUgroup-Gen		Ignored, no exact hostname match in group
(11)	ParmsGroup		Ignored, no exact hostname match in group
(12)	TN3270 (E) Appl		Ignored, no wildcard user ID match in group
(13)	Print Appl		Ignored, no wildcard user ID match in group
(14)	LUgroup-Spec		Ignored, no wildcard user ID match in group
(15)	PRTgroup-Spec		Ignored, no wildcard user ID match in group
(16)	ParmsGroup		Ignored, no wildcard user ID match in group
(17)	Interp table	INTTAB1	Assigned, linkname match
(18)	TN3270(E) Appl	TPX1	Assigned, NULL Client Identifier match
(19)	USS table	USSTAB1	Assigned, NULL Client Identifier match

Mapping methods

Once you have identified all the clients in your network, determine which clients need to remain exact Client Identifiers for special considerations and which clients can be combined or wildcarded into Client Identifier groups. Now map the application and LU Objects to these Client Identifiers.

1. Use LU name mapping statements to assign LU names to connections based on the Client Identifier. This step is required.
2. Use application mapping statements to facilitate session setup based on the Client Identifier.
3. Use the connection parameters mapping statement to change connection parameters based on specific Client Identifiers.
4. Consider the advanced topic features for additional Telnet functions.

LU name mapping statements

Every connection must be represented by an LU name before a session can be initiated. The time of LU assignment depends on the connection type. In general, for TN3270E clients, the LU name is assigned early during connection negotiation before an application name is known. For all other types of clients, the LU name is assigned after application name selection. For details and exceptions to this rule, see “Advanced LU mapping topics” on page 344. Mapping statements define which LU name is assigned to the connection.

DEFAULTLUS: The simplest way to assign LUs is to create a default LU group that Telnet can use for all terminal connections. DEFAULTLUS is a combination statement that defines the LUs in a default group and maps the group to the NULL

Client Identifier. If the client's Client Identifiers do not match any LU mapping statements, the client is identified by the NULL Client Identifier and will be assigned LUs from the default group.

For example, use the following statement to create an LU group with a numeric range of LUG1001 to LUG1100. When Telnet assigns an LU to a terminal connection, it will assign the next available LU from that group of 100 LUs.

```
DEFAULTLUS  LUG1001..LUG1100..FFFFNNN  ENDDEFAULTLUS
```

By default, Telnet uses a sequential selection method to assign LUs from the LU group. No LU name will be reused until all the names in the group have been used. Specifying NOSEQUENTIALLU changes the selection process to always start at the beginning and find the first name available. If the range is large and a large number of LUs are already assigned, NOSEQUENTIALLU might degrade LU lookup performance.

DEFAULTPRT: The DEFAULTPRT statement is used to create a default LU pool that Telnet will use for all printer connections. For example, use the following statement to create an LU group with a numeric range of PRTG1001 to PRTG1100. When Telnet assigns an LU to a printer connection, it will assign the next available LU from that group.

```
DEFAULTPRT  PRTG1001..PRTG1100..FFFFFNNN  ENDDEFAULTPRT
```

LUMAP, PRTMAP, LUGROUP, PRTGROUP: The LUMAP and PRTMAP statements allow you to map LUs to connections based on the Client Identifier for terminal emulators and printer emulators, respectively. For example, use the following statements to map LU name LUT001 to any terminal client identified by the client IP address 1.1.1.1 and map LU name PRT001 to any printer client identified by client IP address 2.2.2.2.

```
LUMAP  LUT001  1.1.1.1
PRTMAP PRT001  2.2.2.2
```

An LU group can be used when it is not necessary to have an exact LU name to Client Identifier match. For example, use the following statements to create a terminal LU group and a printer LU group, and map both groups to the Client Identifier IPGPAY. When a terminal client connects, Telnet will assign an LU from LUGRP1. When a printer client connects, Telnet will assign an LU from PRTGRP1.

```
LUGROUP   LUGRP1    LUT101..LUT400..FFFXXX   ENDLUGROUP
PRTGROUP  PRTGRP1   PRT101..PRT400..FFFXXX   ENDPRTGROUP

IPGROUP IPGPAY   255.255.0.0:9.8.0.0  ENDIPGROUP

LUMAP    LUGRP1   IPGPAY
PRTMAP   PRTGRP1  IPGPAY
```

Once all 300 LUs are assigned, the next client connection request will fail. In this way, the LUGROUP Object can limit the number of clients connected at one time.

If a client connection is known by a Client Identifier that has an LU group mapping, only that mapping will be used to assign an LU name. The DEFAULTLUS group will not be used. It is used only in the case when no other LU mapping exists.

LU range specification: Telnet LU range rules allow for almost any type of LU range needed. Ranges can be alphabetic (A), numeric (N), alphanumeric (B), hexadecimal (X), or completely wildcarded (?) which includes alphanumeric and the three national characters (@,#,$). The range type can be different for each character position. Within the LU range, any character position can be fixed (F). To

conform with VTAM LU naming convention, the first character must be alphabetic or a national character. If the first character is a range, only the alphabetic range can be used.

An LU range is created by specifying a starting LU name, an ending LU name, and the range rules to be used. For example, the following statement creates a range from TCPM1000 to TCPM1100.

```
TCPM1000..TCPM1100..FFFFFNNN
```

The three components are:

- Starting LU name (TCPM1000)
- Ending LU name (TCPM1100)
- Range rules (FFFFFNNN)

All three components must be the same length, the Starting LU name overall must be lower than the Ending LU name, and each character position value must be appropriate for the specified range rule. Notice in the above example that the character 1 following the character M is defined as fixed because it cannot change. The range rule cannot specify N even though it seems to be part of the number range.

The ascending order of characters is 0-9, A-Z, @, #, $.

Numeric values are lower than alphabetic values to facilitate the use of hexadecimal ranges. The range rules are:

```
Range        Rule       Characters
Numeric       N          0-9
Alphabetic    A          A-Z
AlphaNumeric  B          0-9,A-Z
Hexadecimal   X          0-9,A-F
Wildcard      ?          0-9,A-Z,@,#,$
```

The maximum number of LUs per range is 4294967295 and the maximum number of LUs per group is 4294967295.

The creation of LU name values from the range specification begins at the Starting LU and increments the rightmost variable position first, moving to the left as each variable position reaches its range maximum. The process is like an odometer, except that each position can have different basing instead of all positions being base 10. For example, the following statement has 223 LU name entries.

```
LU555..LU777..FFNNN
```

The breakdown of the range is:

```
LU555->LU559,                                                 5
LU560->LU569, LU570->LU579, LU580->LU589, LU590->LU599,      40
LU600->LU699,                                               100
LU700->LU769, LU770->LU777                                   78
                                                            ===
Total  ---------------------------------------------------> 223
```

The LU names increment just as the numbers on an odometer would. A less intuitive case involves an alphabetic range of 1407 LU name entries.

```
LUCCC..LUEEE..FFAAA
```

The breakdown of the range is:

```
LUCCC->LUCCZ,                                                    24
LUCDA->LUCDZ, LUCEA->LUCEZ, LUCFA->LUCFZ, ... LUCZA->LUCZZ,     598
LUDAA->LUDZZ,                                                   676
LUEAA->LUEDZ, LUEEA->LUEEE                                      109
                                                                ====
Total  ------------------------------------------------------> 1407
```

It is important to realize that the ranges above do *not* break down in the following patterns:

```
LUCCC->LUCCE, LUCDC->LUCDE, LUCEC->LUCEE, ...
LU555->LU557, LU565->LU567, LU575->LU577, ...
```

It is an incorrect assumption that the LU name after LUCCE would be LUCDC. The correct LU name after LUCCE is LUCCF. The LU names increment to LUCCZ and the next name is LUCDA. When the rightmost position reaches the range maximum, the position to its left is incremented by one, and the rightmost position starts at the range beginning, not the character specified in the Starting LU name.

All range types are handled the same way. The position is incremented to its maximum value and then wraps to the beginning range value, not the specified Starting LU name value. By the same logic, the position is incremented to the ending range value and not the Ending LU name value.

All LU names increment the same way. A more complicated example mixes fixed and variable character positions with several different range types. The LU range has 39744 LUs.

```
LUAD1800..LUGD98FZ..FFAFNFXB
```

Calculating the number of LUs is easier if the fixed positions are removed. For purposes of calculating the number of LUs, the range is specified as follows:

```
A100..G9FZ..ANXB
```

This breaks down as follows:

```
A100->A10Z, D110->D11Z, ... A190->A19Z, A1A0->A1AZ ... A1F0->A1FZ    576
A200->A2FZ, A300->A3FZ, ... A900->A9FZ                              4608
B000->B9FZ, C000->C9FZ, ... F000->F9FZ                             28800
G000->G9FZ                                                          5760
                                                                   =====
Total ------------------------------------------------------------> 39744
```

SEQUENTIALLU: Telnet, by default, uses a sequential method to choose LUs from a group.

```
LUGROUP  LUGRP1
     LU001..LU120..FFNNN
     LU201..LU250..FFNNN
     LU240..LU280..FFNNN
     LU010..LU050..FFFNN
ENDLUGROUP
```

From the previous example, the first LU assigned is LU001, second is LU002, and so on. If five clients repeatedly connect and disconnect, they will be assigned new LUs farther into the range each time:

- When the end of the first range is reached, selection goes to the beginning of the second range.
- At the end of the second range, selection goes to the beginning of the third range.
- At the end of the third range, selection goes to the beginning of the fourth range.

- At the end of the fourth range, selection goes to the beginning of the first range again.

Telnet does not enforce an overall ascension in LU name selection. The selection process begins at the first name of the first range and progresses to the last name of the last range. In the example, after LU250 is assigned from range 2, LU240 from range 3 is attempted next. After LU280, LU010 is attempted. After LU050, the process starts over and LU001 is attempted.

The SEQUENTIALLU function can be turned off by coding NOSEQUENTIALLU. In this case, the five LUs that are repeatedly connecting and disconnecting would never use any LU names other than LU001, LU002, LU003, LU004, and LU005. NOSEQUENTIALLU might degradate LU lookup performance when a large range is specified and only LUs at the end of the range are available. Every connection has to relearn that most of the LUs are already in use. SEQUENTIALLU allows Telnet to start its search near the last chosen LU where LUs are more likely to be available. SEQUENTIALLU and NOSEQUENTIALLU parameters can be coded at all three parameter block levels for different levels of granularity.

If several clients are connecting at the same time, the order of LU assignment might not be in exactly the same order as the connection IDs due to process timing between connection ID assignment and LU name assignment.

If single LU names are in a group with LU ranges, the single LU names are selected before any LU range names are selected, regardless of their order. In the example below, LUAAA, LUBBB, LUCCC, and LUDDD are all processed before any of the range LU names.

```
 Profile LUGROUP                              LUGROUP as used by Telnet


LUGROUP  LUGRP2                                 LUGROUP  LUGRP2
     LUAAA                                           LUAAA
     LU001..LU120..FFNNN                             LUDDD
     LU201..LU250..FFFNN                             LUBBB
     LUDDD                                           LUCCC
     LUBBB                                           LU001..LU120..FFNNN
     LU240..LU280..FFFNN                             LU201..LU250..FFFNN
     LU010..LU050..FFFNN                             LU240..LU280..FFFNN
     LUCCC                                           LU010..LU050..FFFNN
ENDLUGROUP                                      ENDLUGROUP
```

Application mapping statements

When a client connects, Telnet either immediately initiates a session request to an MVS host VTAM application or solicits the end user for an application name.

DEFAULTAPPL: The DEFAULTAPPL mapping statement is used to assign an application name to the connection and immediately initiate a session with that application, and not solicit the end user for an application name. The DEFAULTAPPL statement applies only to terminal emulators connecting in TN3270, TN3270E, or DBCSTRANSFORM mode. For example, use the following statement to map the default application PAYROLL to any TN3270(E) terminal client identified by the IPGROUP IPGPAY. When a TN3270(E) client connects, Telnet will immediately initiate a session to the PAYROLL application.

```
DEFAULTAPPL PAYROLL IPGPAY
```

PRTDEFAULTAPPL and LINEMODEAPPL: The PRTDEFAULTAPPL mapping statement is used to assign an application to a printer emulator client connecting in TN3270E mode. The LINEMODEAPPL mapping statement is used to assign an

application to a client connecting in standard or binary LINE mode. For example, use the following statements to map the default application PAYPRINT to any TN3270E printer client identified by the IPGROUP IPGPAY and to map the default application TSO to any linemode client identified by the linkname LINK1. When the printer client connects, Telnet will immediately initiate a session to the PAYPRINT application. When a linemode client connects, Telnet will immediately initiate a session to the TSO application.

```
PRTDEFAULTAPPL PAYPRINT IPGPAY
LINEMODEAPPL  TSO  LINK1
```

The DEFAULTAPPL, PRTDEFAULTAPPL, and LINEMODEAPPL statements imply a basic ALLOWAPPL statement for the application name if no ALLOWAPPL or RESTRICTAPPL is explicitly coded.

USSTCP: If the end user needs the ability to choose an application, custom solicitation panels can be created using unformatted system services (USS) message tables. These tables are mapped to clients using the USSTCP mapping statement. For example, use the following statement to map a USS table, USSTAB1, to any TN3270(E) client identified by any linkname that starts with LINK. When a TN3270(E) client connects, Telnet will immediately send a custom logon screen (USSMSG10) from the USS table.

```
LINKGROUP  LNKGRP1  LINK*   ENDLINKGROUP
USSTCP USSTAB1 LINKGRP1
```

Assembled USS tables used by VTAM can also be used by Telnet.

INTERPTCP: In some cases, the application name must be generated based on the name provided by the end user or the name might be dependent on the LU name representing the client. The INTERPRET table can provide this function. Telnet uses the input from the USSMSG10 screen as input to the INTERPRET table translation list or uses the USSMSG10 input and the LU name as input to one of the INTERPRET table user-written exits. Because USS logon data is required input to the INTERPRET process, any client with an INTERPRET table mapping must also have a USS table mapping. For example, use the following statement to map an INTERPRET table, INTTAB1, to any TN3270(E) client identified by the linkname LINK1. When a TN3270(E) client connects to LINK1, Telnet will immediately send a custom logon screen (USSMSG10) from the USS table. The end user responds with a USS logon command. LINK1 client input is then processed through the INTTAB1 INTERPRET table to derive an application name. Telnet uses the derived name to initiate a session.

```
LINKGROUP LNKGRP1
     LINK*
ENDLINKGROUP
USSTCP    USSTAB1 LNKGRP1
INTERPTCP INTTAB1 LINK1
```

Assembled interpret tables used by VTAM can also be used by Telnet.

If neither a default application nor a USS table is mapped to the connection, the Telnet Solicitor panel is sent to the end user. For a detailed discussion of the Telnet Solicitor, USS table, and INTERPRET table, see “Using the Telnet Solicitor or USS logon panel” on page 362.

Resolving DEFAULTAPPL and USS table conflicts: If both a default application and a USS table are mapped to the same Client Identifier, Telnet will use the default application to immediately initiate a session. If each is mapped by a different Client Identifier, the Object mapped by the higher priority Client Identifier is used. In

all cases, any error messages are sent using the USS table messages. For example, if CICS and USSTAB1 are both mapped to destination IP address 1.1.1.1, Telnet will initiate a session with CICS and use the USS messages for any session setup errors.

If CICS is mapped to USERID USER1 and USSTAB1 is mapped to client IP address 5.5.5.5, Telnet will initiate a session with CICS and use the USS messages for any session setup errors.

If CICS is mapped to linkname LINK1 and USSTAB1 is mapped to hostname TEST1.IBM.COM, Telnet will send a USSMSG10 logon panel to the end user. The USS messages will be used for any session setup errors. The default application mapping of CICS will never be used.

ALLOWAPPL: Telnet will not initiate a session for a solicited application name unless the name is allowed. The ALLOWAPPL statement is used to configure Telnet to allow the initiation request. For example, CICS01 and CICS02 are allowable names.

```
ALLOWAPPL CICS01
ALLOWAPPL CICS02
```

The ALLOWAPPL name can be wildcarded with an asterisk (*). For example, if there are no other CICS regions, the lines above could be reduced to the following:

```
ALLOWAPPL CICS*
```

All application names can be allowed by coding the following:

```
ALLOWAPPL *
```

Default application names do not need to be explicitly allowed. However, if the default application issues a CLSDST-PASS to another application name for the session, the second application must be in the ALLOWAPPL list. For example, TSO is the default application for the NULL Client Identifier. TSO typically passes the session to TSO00001, TSO00002, and so on. The following default application mapping will initiate a session with TSO, but when TSO issues a CLSDST-PASS the new bind to Telnet will have TSO00001 as the application name.

```
DEFAULTAPPL TSO
```

Telnet will fail this session request because TSO00001 is not allowed. Add an ALLOWAPPL statement to allow the TSO* names as follows:

```
DEFAULTAPPL TSO
ALLOWAPPL   TSO*
```

RESTRICTAPPL: In addition to the ALLOWAPPL statement, Telnet provides more restrictive access to applications. The RESTRICTAPPL statement requires the end user to enter a valid RACF user ID and password before the application name is used to initiate a session.

For example, use the following statement to allow users USER1, USER2, USER3, USER4, and USER5 access to the PAYROLL application. At the Solicitor panel, the end user enters USER1/password and the PAYROLL application name. Telnet verifies USER1/password is valid and then immediately initiates a session with PAYROLL.

```
RESTRICTAPPL PAYROLL
   USER USER1
   USER USER2
   USER USER3
   USER USER4
   USER USER5
```

Like ALLOWAPPL, the application name can be wildcarded with an asterisk (*). The USER value can also be wildcarded with an asterisk. The user ID/password combination is used by Telnet to verify the password given for that user ID. In no way is the user ID or password used by the application. No matter how the application name request arrived at the server (from DEFAULTAPPL or USSMSG10), Telnet uses the Solicitor panel to prompt for the user ID/password. Once the user ID is validated and a password is obtained, Telnet submits the user ID/password pair for authorization to a security program such as RACF. The user ID/password check authorizes the client to connect to the application through Telnet. The application itself might also ask for a user ID/password pair that can be completely different than the pair entered at the Telnet Solicitor panel. The user ID/password pair entered at the Telnet Solicitor panel is not in any way passed to the host application. The user ID/password pair is solicited only after an application name is entered on the Solicitor (or USSMSG10) panel. If a second application is reached through the original application using CLSDST-PASS, the second application is verified and Telnet will solicit a new user ID/password pair if necessary.

When searching for a match with the input application name, Telnet will find the most specific match whether it is on the ALLOWAPPL or RESTRICTAPPL statement. If each statement has the same name specified, the RESTRICTAPPL entry is used. For example, TSO has its own user ID/password requirement and probably does not need the additional Telnet security check. However, the Telnet security check may be needed for all other applications. This example can be supported with the following statements.

```
RESTRICTAPPL   *
  USER   *
ALLOWAPPL   TSO*
```

MSG07 and LUSESSIONPEND: MSG07 and LUSESSIONPEND are Telnet parameter statements that define what Telnet should do in case of a session setup error and after normal logoff when the client is emulating a terminal. These parameters do not affect a printer connection.

- Connection negotiation error - If any problems occur during negotiation nothing can be done to keep the connection. If appropriate, Telnet will send the client an error code to help inform the client why the connection was dropped and issue a CONN DROP DEBUG message at the console.
- Session setup error - If a problem occurs during session setup such as an application name that is not valid, session request failure, or a BIND error, Telnet will drop the connection and issue a CONN DROP DEBUG message. The end user cannot get to any application other than the default. No error messages are sent to the end user and auto-reconnect loops are possible. For these reasons it is recommended that MSG07 always be used. If the MSG07 parameter is coded, the connection will not be dropped and an error message will be sent to the end user. MSG07 function applies to any connection mode whether or not USS tables are mapped to the client. If a USS table is used, the end user can press the CLEAR key to return to the USSMSG10 screen. If the LUMAP-DEFAPPL or PRTMAP-DEFAPPL statement is coded and the default application is not available, an error screen will be sent to the client whether or not MSG07 is coded.

- Normal Session Logoff - When the end user logs off a session using a normal logoff, Telnet drops the connection. If the end user typically logs on to another application after logging off the first application, it might be more efficient if the user were presented another solicitor (or USSMSG10) panel or if Telnet initiated a new session with the default application after logoff. This can be accomplished by coding the LUSESSIONPEND parameter. Code LUSESSIONPEND to redrive the initial database lookup after session logoff. Later results will be identical to the first lookup. If a default application for the client exists, Telnet will immediately initiate another session request. Otherwise, a USSMSG10 screen or solicitor panel will be sent to the end user. When LUSESSIONPEND is coded, the connection remains active but terminal LU ACBs are closed.
- SYSREQ LOGOFF - When the end user logs off a session using a ″SYSREQ LOGOFF″ sequence (TN3270E connection supported) and LUSESSIONPEND is coded, Telnet does not drop the connection. Instead, the user is presented with a solicitor (or USSMSG10) panel. If DEFAULTAPPL is in effect, Telnet redrives the default application.
- USS LOGOFF - When the end user issues a LOGOFF command from the USSMSG10 panel, the connection is dropped whether or not the LUSESSIONPEND parameter is coded.

Connection parameters mapping statement

Connection parameters are typically defined once at the port level. Sometimes it is useful to have different connection parameters depending on the Client Identifier. The PARMSGROUP and PARMSMAP statements allow connection parameters to be mapped at the Client Identifier level. This level of granularity applies to almost all parameters. Refer to the *z/OS Communications Server: IP Configuration Reference* for a list of Telnet parameters allowed in the PARMSGROUP block.

Assume the PAYROLL department is assigned the highest level of security and connections are being monitored with summary debug messages, general users are assigned negotiable security, and inventory employees are experiencing intermittent problems with Telnet connections that require detailed debug messages for resolution. The following statements assign the security and debug levels to the areas needed and do not affect other areas. See “Transport layer security” on page 320 for security information and “Telnet diagnostics” on page 367 for debug information.

```
HNGROUP   HNGINV
  **.GROUP3.COM
ENDHNGROUP
IPGROUP   IPGPAY
  255.255.0.0:2.2.0.0
ENDIPGROUP
IPGROUP   IPGGEN
  255.0.0.0:2.0.0.0
ENDIPGROUP
PARMSGROUP   PRMGDBG
  DEBUG DETAIL
ENDPARMSGROUP
PARMSGROUP   PRMGSEC1
  CONNTYPE SECURE
  ENCRYPTION SSL_3DES_SHA  ENDENCRYPTION
  DEBUG SUMMARY
ENDPARMSGROUP
PARMSGROUP   PRMGSEC2
  CONNTYPE NEGT
  ENCRYPTION SSL_RC4_MD5   ENDENCRYPTION
```

```
ENDPARMSGROUP
PARMSMAP   PRMGDBG  HNGINV
PARMSMAP   PRMGSEC1 IPGPAY
PARMSMAP   PRMGSEC2 IPGGEN
```

Advanced LU mapping topics

Beyond the basic LU mapping statements, there are several functions available to the advanced user. This section describes the following topics:

- Generic and Specific connection requests
- Mapping groups to Client Identifiers
- LU name assignment user exit
- Associated printer function
- Map default application and ParmsGroup by LU group
- Multiple LUMAP statements for one Client Identifier
- Keep LU for the Client Identifier
- LU group capacity warning
- LU mapping by application name
- LU mapping selection rules

Generic and Specific connection requests

There are three types of Telnet connection requests that dictate how the Telnet server chooses a name to represent the client. They are generic requests, specific requests, and associated printer requests. For details about associated printer requests, see “Associated printer function” on page 347. Most connection requests are generic requests.

For Generic requests, the Telnet server has complete control over LU name assignment using the Generic mapping statements as a reference. All linemode and TN3270 connections use Generic requests, and TN3270E terminal and printer emulators use Generic requests as a default. Specific mapping statements are ignored by Generic requests.

For Specific requests, the client specifies the LU name to be used and the Telnet server validates the name using the Specific mapping statements as a reference. Requesting a specific LU name allows a client to be assigned the same LU every time. This is important if the host application is LU name dependent, and the client does not have a constant Client Identifier to use for mapping an LU name. It is also important to block the server from assigning these LUs to generic requests. If a Specific mapping does not find an LU match, Generic mapping statements are checked. The server confirms or denies the request during negotiation. If the LU mapping algorithms reject the client choice, the server sends a device type reject to the client. Most clients then notify the end user that the requested LU name is not valid or is already in use.

Default LU groups: DEFAULTLUS and DEFAULTPRT are default LU groups for generic requests. DEFAULTLUSSPEC and DEFAULTPRTSPEC are default LU groups for specific requests from terminal and printer emulators. Like DEFAULTLUS and DEFAULTPRT, these pools are checked only if there is no other LU mapping statement match. For example, use the following statements to create a terminal LU group with a numeric range of LUS1001 to LUS1100 and a printer LU group with a numeric range of PRTS1001 to PRTS1100. When Telnet receives a specific connection request from a terminal, it will verify that the requested LU name is within the range specified.

```
DEFAULTLUSSPEC  LUS1001..LUS1100..FFFFNNN  ENDDEFAULTLUSSPEC
DEFAULTPRTSPEC  PRTS1001..PRTS1100..FFFFFNNN  ENDDEFAULTPRTSPEC
```

The sequential selection rules do not apply to specific requests.

Mapping groups to Client Identifiers

The LUMAP and PRTMAP statements allow LUs to be mapped based on a Client Identifier. The LU group can be mapped generically or specifically. The default mapping is generic. The keyword SPECIFIC must be coded to define a specific mapping.

For example, use the following statements to create two LU groups and map one group generically to the IP group IPGPAY and map the other group specifically to the same Client Identifier. When a generic connection request is received, Telnet will assign the next available LU from LU group LUGRPGEN. When a specific connection request is received, Telnet will verify the requested LU name is included in the LU group LUGRPSPC. If it is, Telnet will assign the LU name to the connection.

```
LUGROUP   LUGRPGEN  LUG101..LUG400..FFFXXX   ENDLUGROUP
LUGROUP   LUGRPSPC  LUS001..LUS100..FFFXXX   ENDLUGROUP

IPGROUP IPGPAY   255.255.0.0:9.8.0.0  ENDIPGROUP

LUMAP    LUGRPGEN   IPGPAY
LUMAP    LUGRPSPC   IPGPAY  SPECIFIC
```

Generic request connections can be assigned LUs only from generically mapped LU groups. If no generic mapping exists, the DEFAULTLUS group is checked. No specific group is checked. This safeguards the specific LU names from being used by generic requests.

For specific requests, Telnet first checks to see if the LU is in a specifically mapped LU group. If the LU name is not found, the generically mapped groups are searched. If neither LU group type contains the requested LU name, the connection request is rejected. The DEFAULTLUSSPEC group is *not* checked in this case because LU group mappings exist. If no LU group mappings exist, only the DEFAULTLUSSPEC group is checked. If the LU name is not found, the connection request is rejected. The generic DEFAULTLUS group is *not* checked.

In addition to requesting an exact LU name, the client can request an LU group name. Telnet first searches the mapped groups assuming the name is an exact LU name. When that search fails, Telnet then checks the requested name against mapped specific LU group names and then checks the name against mapped generic LU group names. If the group name is found, the next available LU in the group is assigned using sequential LU selection unless it has been turned off.

The LU group itself is not defined as generic or specific. Rather, the LU group is *mapped* generically or specifically. It is possible to map the same LU group both generically and specifically. IBM recommends that you do not map the same group generically and specifically unless you are an advanced user.

LU name assignment user exit

Most LU assignment requirements can be satisfied using the Telnet LU group and LU mapping statements. However, there are cases when the LU assignment requirements are so specific that Telnet can not satisfy them. In these cases, the LU name assignment user exit might be the solution. The LU name exit is defined like an LUGROUP and is mapped the same way LUGROUPs are mapped. The LUGROUP is defined as an exit by specifying *,EXIT* immediately after the

LUGROUP name. For LUGROUPs, Telnet selects an LU from the group, verifies its availability, and assigns the LU to the connection. For LU name exits, Telnet calls the user-written assembler program passing a parameter list that contains client and other information. The program creates the LU name, places it in the parameter list, and returns control to Telnet. Telnet will then verify the LU name's availability and assign the LU to the connection. For a detailed description of the parameter list and coding requirements for the Telnet LU Exit, refer to *z/OS Communications Server: IP Configuration Reference*.

In addition to client information, the parameter list includes any LU names or ranges that were coded in the LUGROUP, and the requested application name if known. Telnet does not use the LU list. The LUGROUP can be defined without any LUs specified. The LUs specified can be used as seed values if the LU name exit wants to use them.

The LU name exit is called when the LU is assigned, when the LU is released, when the LU is inactivated, and when the LU is activated. A different function code is used for each type of call. If you do not need a certain function, like tracking inactivated LUs, the LU name exit can be written to ignore the function code. Telnet allows only one connection at a time to use the LU name exit, which serializes its use in case any local tables are maintained in the exit.

As an example, assume LU names are to be assigned based on client port number and application requested. The SIMCLIENTLU parameter is used to postpone TN3270E LU assignment until the application name is known. The parameter list includes the client port number and the requested application name. In this case, no seed LU names are needed. The LU name exit will create LU names based on the port number and application name in the parameter list.

```
LUGROUP  LUEXIT1,EXIT
ENDLUGROUP
```

In another example, assume the clients specify LUGROUP names that match up with default applications on the LUMAP statement. The LU names are to be created based on the last two numbers of the IP address and a prefix that identifies the application. For example, TSO, IMS, and CICS are three current applications. The prefix for each is TS, IM, and CI, respectively. The connection from 9.1.240.111 specifies LUGROUP LUGTSO and is assigned TS240111. The connection from 9.1.240.212 specifies LUGROUP LUGIMS and is assigned IM240212. The connection from 9.1.89.7 specifies LUGROUP LUGTSO and is assigned TS089007. Three LU name exits are needed (LUGTSO, LUGIMS, LUGCICS), but they are all functionally equivalent. The LU name specified in the LUGROUP is passed to the LU name exit as part of the parameter list. That name is used by the exit as the prefix. The client IP address is also in the parameter list. The LU name exit combines the prefix with the last portions of the IP address to create an LU name. The following statements can be used to support this scenario.

```
IPGROUP  IPGRP1  0.0.0.0:0.0.0.0  ENDIPGROUP  ; Matches all connections


LUGROUP  LUGTSO,EXIT    TS       ENDLUGROUP
LUGROUP  LUGIMS,EXIT    IM       ENDLUGROUP
LUGROUP  LUGCICS,EXIT   CI       ENDLUGROUP

LUMAP  LUGTSO  IPGRP1  DEFAPPL  TSO
LUMAP  LUGIMS  IPGRP1  DEFAPPL  IMS
LUMAP  LUGCICS IPGRP1  DEFAPPL  CICS
```

Capacity checks cannot be performed since Telnet has no way of knowing how many total LUs are available in the LU name exit.

Associated printer function

The associated printer function allows a printer to specify an active LU terminal name during connection negotiation. The server understands this special request and knows to assign a printer LU name that is associated with the requested terminal LU name. The association is established by linking a pool of terminal LUs (LUGROUP) with a pool of printer LUs (PRTGROUP). The groups are linked with the LUMAP statement. The printer LU group name is linked to the terminal LU group name by adding the PRTGROUP name on the LUMAP statement.

The two LU groups *must* have the same number of LUs defined so the LUs can be paired. The groups must have the same number of single LU names, the same number of LU ranges, and the same number of LU names in each range. If the groups do not have the same number of LUs defined, error messages will be produced during profile processing and during associated connect requests.

Once the groups are linked, Telnet assigns the *n*th printer LU to a printer connection that requests association with the *n*th terminal LU. For example, a CICS table might specify that if terminal LU1 is requesting printer function, the output should be routed to printer PRT1. Within CICS, LU1 and PRT1 are associated with each other. Use the following statements to set up printer association.

```
LUGROUP  LUGCICS   LU1..LU9           ENDLUGROUP
PRTGROUP PRTCICS   PRT1..PRT9         ENDPRTGROUP
IPGROUP  IPGRP9    255.0.0.0:9.0.0.0  ENDIPGROUP
LUMAP  LUGCICS  IPGRP9  GENERIC  PRTCICS
```

Neither the LU group nor the printer group can be an LU exit group. If either is an LU exit, the mapping statement will be rejected.

Drop the printer connection when dropping the terminal connection: In many cases, the associated printer connection should be dropped when the terminal connection is dropped. If you code the DROPASSOCPRINTER parameter, Telnet will monitor the terminal connection. When the terminal connection is dropped, Telnet will initiate the closing and dropping of the printer connection. The DROPASSOCPRINTER and NODROPASSOCPRINTER parameters can be coded at all three parameter block levels for different levels of granularity.

Map default application and ParmsGroup by LU group

The DEFAPPL option on the LUMAP statement allows a host VTAM application to be mapped with an LU name or LUGROUP name instead of using DEFAULTAPPL. The LUMAP-DEFAPPL combination is treated just like DEFAULTAPPL when a Client Identifier matches the LUMAP statement. The LUMAP-DEFAPPL combination also supports the LOGAPPL, FIRSTONLY, and DEFONLY parameters that are used by DEFAULTAPPL, PRTDEFAULTAPPL, and LINEMODEAPPL. The LUMAP-DEFAPPL combination is a powerful statement when used with multiple LUMAP statements for the same Client Identifier. If the LUMAP-DEFAPPL or PRTMAP-DEFAPPL statement is coded and the default application is not available, an error screen will be sent to the client whether or not MSG07 is coded.

The PMAP option on the LUMAP statement allows assignment of connection parameters based on LU or LU group name. When the LU is assigned, the parameter values specified in the PMAP PARMSGROUP will override the parameter value specified in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP mapped to this connection's Client Identifier. For example, any client residing in subnet 9.0.0.0 that specifies the LU group LUGTSO will immediately have a session

initiated to TSO with the LOGAPPL function, and the TIMEMARK time will be set for two hours instead of the default three hours.

```
IPGROUP    IPGRP9  255.0.0.0:9.0.0.0             ENDIPGROUP
LUGROUP    LUGTSO  TCPTSO01..TCPTSO99..FFFFFFNN  ENDLUGROUP
PARMSGROUP PGRPT2  TIMEMARK 7200                 ENDPARMSGROUP
LUMAP     LUGTSO  IPGRP9  DEFAPPL TSO LOGAPPL  PMAP PGRPT2
```

An LUMAP-DEFAPPL defined default application name is always used if specified, regardless of USSTCP mappings. LUMAP-DEFAPPL has a higher priority than any DEFAULTAPPL or USSTCP. The connection parameters assigned using LUMAP-PMAP will override any other setting of the parameters. However, not all parameters have a meaningful use by the time the LU is assigned. For example, NOTN3270E controls whether or not Telnet should negotiate for TN3270E. That negotiation is done before LU assignments. For information on which parameters can be properly applied with LUMAP-PMAP, see the parameter table in *z/OS Communications Server: IP Configuration Reference*.

The PRTMAP statement supports PRTMAP-DEFAPPL and PRTMAP-PMAP in the same manner as LUMAP-DEFAPPL and LUMAP-PMAP.

Multiple LUMAP statements

Another feature of Specific LU name requests is that the client can specify an LUGROUP name, and the server will assign an available LU from that pool. This capability is useful when different applications require different LU naming schemes, but each end user client does not need to use the exact same LU name for each connection. For example, an administrator can create three pools, one for each of three applications. Only three client emulators need to be set up. One for TSO which requests LU name LUTSO, one for CICS which requests LUCICS, and one for IMS which requests LUIMS. Assume the general users are in subnet 3.0.0.0. Any client connecting with a Client Identifier of IPGGEN can be set up to issue a Specific request for LU pool LUTSO, LUCICS, or LUIMS, and will be assigned an LU from the appropriate pool.

After an LU is assigned, the DEFAPPL option will cause Telnet to immediately issue a session request for the appropriate application. If LOGAPPL is coded and the application is not active, VTAM will continue session initiation once the application is active.

In most cases, DEFAPPL on multiple Generic LUMAPs is not useful. LUs are assigned in order starting with the first LUMAP statement. One case that may be useful is if an application has a user limit but can be cloned. Assume the INVENTRY application can support only 20 users but can be cloned. Multiple LUMAPs with DEFAPPL will direct the first 20 HNGINV clients to INVENTRY, the next 20 HNGINV clients to INVENTR2, and the next 20 HNGINV clients to INVENTR3.

```
IPGROUP  IPGGEN
     255.0.0.0:3.0.0.0
ENDIPGROUP
LUGROUP  LUTSO
     TSO00001..TSO00999
ENDLUGROUP
LUGROUP  LUCICS
     CICS0001..CICS0999
ENDLUGROUP
 LUGROUP  LUIMS
     IMS00001..IMS00999
ENDLUGROUP
LUGROUP   LUGINV1
  LUINV01..LUINV20
```

```
ENDLUGROUP
LUGROUP   LUGINV2
  LUINV21..LUINV40
ENDLUGROUP
LUGROUP   LUGINV3
  LUINV41..LUINV60
ENDLUGROUP
LUMAP   LUTSO     IPGGEN   SPECIFIC  DEFAPPL  TSO      LOGAPPL  FIRSTONLY
LUMAP   LUCICS    IPGGEN   SPECIFIC  DEFAPPL  CICS     LOGAPPL
LUMAP   LUIMS     IPGGEN   SPECIFIC  DEFAPPL  IMS      LOGAPPL
LUMAP   LUGINV1   HNGINV             DEFAPPL  INVENTRY LOGAPPL  DEFONLY
LUMAP   LUGINV2   HNGINV             DEFAPPL  INVENTR2 LOGAPPL  DEFONLY
LUMAP   LUGINV3   HNGINV             DEFAPPL  INVENTR3 LOGAPPL  DEFONLY
```

Pool name specification is a powerful mapping method because multiple LUMAP statements with different Objects can be used for a single Client Identifier.

Keep LU for the Client Identifier

An LU name can be kept (or reserved) for a period of time so no other client is assigned that name. Only the same Client Identifier reconnecting to Telnet within the specified time can be assigned that LU name. After the specified time, the LU name is again available for any connection. This function is useful when the application does not clean up session information quickly and a released LU is quickly reassigned to another end user by Telnet. The application thinks the new session is a continuation of the previous session but it is not. With KEEPLU, the LU will not be reassigned to a different Client Identifier for a period of time, long enough for the application to clean up its session information. The LU name is kept based on the highest Client Identifier by which the connection is known. It is either a User ID, a Hostname, or an IP address, respectively.

LU group capacity warning

An LU group capacity threshold can be specified on the LU group statement. If it is, Telnet will check the number of LUs used in the group when an LU is assigned from the group. A message is issued when the group's in-use LU count is at or above the specified percentage of the total. Once the message is issued, no other message is issued until the in-use count has dropped below the threshold by 10% of the total. For example, an LU group has 200 LUs with a capacity threshold of 80%. When the 160th LU is assigned, EZZ6007I is issued. Ten percent of the total is 20. Therefore, after the number of LUs has dropped to 140 or lower, another warning message will be issued when the in-use count rises to 160 again. If multiple LU groups have the same LU name, the only LU group checked is the group mapped to the client being assigned the LU. The other LU groups might be over their capacity amounts but notification will not be issued until an LU is taken from the group. Below are examples for setting the capacity warning.

```
LUGROUP   LUGRP1,80%    TCPLU000..TCPLUF9F..FFFFFXNX
PRTGROUP  PRTGRP1,60%   TCPRT000..TCPRTFFF..FFFFFXXX
```

Capacity checking cannot be done for LU groups that are defined as LU naming exits. During VARY TCPIP,,OBEYFILE processing, all LU groups are checked for in-use LU counts and a capacity warning message is issued if needed.

LU mapping by application name

In some cases, only certain LU names are eligible to be in session with the host application. Or only certain LU names are eligible to represent user IDs. The LU and LUG parameters on the ALLOWAPPL and RESTRICTAPPL statements provide this checking function and allow some LU name mapping based on application name.

For example, assume the only LUs eligible to use the inventory set of applications are the LUs in the inventory LU pools. A new LUGROUP pool named LUGINVT contains LUs from LUGINV1, LUGINV2, and LUGINV3. The ALLOWAPPL statement requires that any session request to the inventory applications have an LU name defined in LUGINVT. The LUG parameter must be used carefully. When specified, Telnet must match the LU using both the common mapping algorithms and the mapping by application. For RESTRICTAPPL, assume security authorization is required to get to the PAYROLL application, and each of the PAYxx user IDs must map to a certain LU.

```
LUGROUP   LUGINV1
  LUINV01..LUINV20
ENDLUGROUP
LUGROUP   LUGINV2
  LUINV21..LUINV40
ENDLUGROUP
LUGROUP   LUGINV3
  LUINV41..LUINV60
ENDLUGROUP
LUGROUP   LUGINVT
  LUINV01..LUINV60
ENDLUGROUP
ALLOWAPPL   INVENTR*  LUG   LUGINVT
RESTRICTAPPL   PAYROLL
  USER  PAY01     LU    LUPAY01
  USER  PAY02     LU    LUPAY02
        (user pay03 through pay20 not listed)
```

The LU group specified on the LUG parameter cannot be an LU exit. If it is, the ALLOWAPPL statement is rejected. Multiple LUs can be assigned individually using the LU keyword or a single LU group can be assigned using the LUG parameter. LU and LUG cannot be mixed on a single statement and only one LUG entry per statement is permitted. LU assignment based on application is a convenient way to limit the access to applications. However, this increases mapping complexity significantly when LU mapping statements and connection types are part of the overall mapping equation. Non-TN3270E connections or TN3270E connections with NOTN3270E or SIMCLIENTLU specified do not keep the LU name assigned to the connection after a session is dropped. For these connection types, the end user can establish a session with different application names even if different LU names are mapped to the application names with the ALLOWAPPL or RESTRICTAPPL-USER statement. However, LU mapping based on application name does not work well with TN3270E connections because the LU is assigned during connection negotiation before the desired application name is known. In all CLSDST-PASS cases, the LU name cannot change when switching from the first application to the second because the LU's ACB is not closed during the switch. If the LU mapping by application name requires an LU name switch, the new session attempt will be failed by Telnet.

TN3270 connections do not assign an LU to represent the client until an application name is chosen. Therefore, the LU and LUG parameters can be used as sole LU mapping statements for TN3270 connections. For example, assume no other mapping statements exist (LUMAP or DEFAULTLUS), and either no TN3270E connections will be used or SIMCLIENTLU has been specified. The following ALLOWAPPL statements will map LUs to the appropriate application based on the application name chosen. The following RESTRICTAPPL statement will assign a single LU or LU pool to each user.

```
ALLOWAPPL   TSO*  LUG   LUGTSO
ALLOWAPPL   CICS  LUG   LUGCICS
ALLOWAPPL   IMS   LUG   LUGIMS
```

```
RESTRICTAPPL   APP*
  USER   USER01   LUG  LUG01
  USER   USER02   LU   LU02
  USER   USER03   LU   LU03
```

Both of these assignment methods were very popular before TN3270E connections were introduced. TN3270E connections will likely achieve poor mapping results. An LU must be assigned during connection negotiation before the application name is known which will likely result in an LU mismatch later. TN3270E connections require that an LU mapping statement exist because an LU must be assigned to the connection during negotiations before an application name is known. Consider the following example:

```
DEFAULTLUS
  LU1 LU2 LU3 LU4
ENDDEFALTLUS
RESTRICTAPPL  APPL1
  USER  USER3  LU  LU3
ALLOWAPPL   APPL2  LU  LU4
```

Assume two TN3270 connections are started.

- Two solicitor screens appear.
- Specify APPL1, USER3, and a password. The server selects LU3 based on both the DEFAULTLUS and the RESTRICTAPPL statements.
- Specify APPL2. The server selects LU4 based on both the DEFAULTLUS and the ALLOWAPPL statements.

Assume two TN3270E connections are started.

- Two solicitor screens appear. The server assigns LU1 and LU2.
- Specify APPL1, USER3, and a password. The server fails the connection because of an LU mismatch.
- Specify APPL2. The server fails the connection because of an LU mismatch.

If LU name mapping by application name or user ID is desired with TN3270E clients, the following three solutions are available:

- If the same application or user ID is always used at the same client, individual LUMAP statements can be used to map the correct LU name to each client. Then every connection request will result in the correct LU assignment for that client. The assumptions are that the client keeps the same Client Identifier and only one client exists per Client Identifier.
- Map the NOTN3270E parameter to clients to disable all TN3270E function in the server so those connections will be TN3270, not TN3270E. The drawback is that all TN3270E function is disabled. This includes printer function, Generic/Specific function, and SNA function to the client. The TN3270E and NOTN3270E parameters can be coded at all three parameter block levels for different levels of granularity.
- Mapping the SIMCLIENTLU parameter is a less severe solution. This function will send a dummy LU name of EZBSIMLU to all TN3270E clients issuing Generic connection requests to satisfy the negotiation but will not assign a Telnet LU until an application name is chosen. This alternative preserves printer function, Specific requests, and SNA function to the client. The drawback is the name sent to the client is not the name Telnet ultimately uses to represent the client. Printer association will not work for these TN3270E Generic connections and any emulator programming that depends on the LU name will be using the dummy LU name. The SIMCLIENTLU and NOSIMCLIENTLU parameters can be coded at all three parameter block levels for different levels of granularity.

LU mapping selection rules

LU mapping selection can become complicated because of the many variations of mapping statements, TN3270E versus TN3270 connections, Generic versus Specific connection requests, printer association, and LU mappings based on application name. LU mapping is very different between TN3270E and TN3270 and will be discussed separately. But first, some general Mapping Rules for both TN3270E and TN3270 follow:

- If multiple LUMAP statements exist for a Client Identifier all Specific LUMAPs are searched (TN3270E only) and then all Generic LUMAPs are searched in the order they are listed in the profile.
- If the application is known during the LU lookup and the ALLOWAPPL or RESTRICTAPPL-USER statement has LUs listed, then the found LU must be in both the mapped LU group and in the application LU group.
- Once an LU match is found, the search stops.
- Telnet performs database lookup for Objects based on the Client Identifier. TN3270E connections require an Early Lookup so Telnet can give the client an LU name during connection negotiation. In all cases a Complete Lookup is done when the application name is given. Telnet performs an Early Lookup and a Complete Lookup for TN3270E connections. Telnet performs only a Complete Lookup for TN3270 and Linemode connections.

TN3270E LU mapping: TN3270E connections require an Early Lookup during connection negotiation. Telnet will use as much information as is available to assign an LU to the client. However, the eventual application is not known at this time unless an LUMAP-DEFAPPL or DEFAULTAPPL statement defines the application name. After connection negotiations are complete, Telnet will either send a logon solicitor (or USSMSG10) screen to the client or will perform a Complete Lookup using the application name obtained from the LUMAP-DEFAPPL or DEFAULTAPPL statement. If Complete Lookup is successful, Telnet will begin session initiation. If a solicitor (or USSMSG10) screen is sent to the client, an application name must be entered, at which time Telnet will perform a Complete Lookup. If LU mapping is being done based on application name, a conflict might occur between the application LU mapping and the LU already assigned to the connection. For TN3270E, once an LU name is assigned during connection negotiation it can never change until the connection is dropped. The SIMCLIENTLU statement allows Telnet to assign LUs for TN3270E connections as though they were TN3270 connections. See “TN3270 LU mapping” on page 353 for mapping Generic TN3270E connection requests with SIMCLIENTLU. A request for a specific LU from the Telnet Client will be treated as if SIMCLIENTLU were not specified. The exact lookup process for TN3270E (non-SIMCLIENTLU) is described below.

Early Lookup: An LU must be found during Early Lookup. LUMAP-DEFAPPL and DEFAULTAPPL statements are considered but not necessarily used. Possible lookup results are:

- An LU is found.
- An LU is not found, the connection is dropped.

Perform TN3270E Early Lookup in the following order. The process stops when LU lookup is successful. Printer connections use the same process, substituting PRTMAP and PRTDEFAULTAPPL.

- Check for LUMAP matches considering application lookup results and possible application-based LU mappings.

1. For each Specific LUMAP used for Specific connection requests: If the Specific LUMAP has DEFAPPL, or DEFAULTAPPL was specified and the application lookup return code is either OK or USER_REQUIRED, then perform LU lookup.
2. For each Generic LUMAP used for Specific or Generic connection requests: If the Generic LUMAP has DEFAPPL, or DEFAULTAPPL was specified and the application lookup return code is either OK or USER_REQUIRED, then perform LU lookup.

- Check for LUMAP matches without considering application lookup results.
  1. For each Specific LUMAP used for Specific connection requests: Ignore DEFAPPL and DEFAULTAPPL and perform LU lookup.
  2. For each Generic LUMAP used for Specific and Generic connection requests: Ignore DEFAPPL and DEFAULTAPPL and perform LU lookup.
- If LUMAP statements were *not checked* (different from *checked but no match*), use the appropriate Default LU pool considering application lookup results and possible application-based LU mappings. In this case the only relevant application is the DEFAULTAPPL, if specified. If the application lookup return code is either OK or USER_REQUIRED, then perform LU lookup.
- If no LUMAP statements were checked, try the appropriate Default LU pool without considering application lookup results. Perform LU lookup.

Complete Lookup: An application name is required for Complete Lookup. The application name is obtained from one of three sources in the order specified.

1. Input from the USER or VTAM (via CLSDST with OPTCD=PASS)
2. DEFAPPL parameter on the LUMAP statement
3. DEFAULTAPPL statement

Use the application name and the previously found LU to perform Complete Lookup. Possible lookup results are:

- The application is not valid.
- The application is valid (return code OK or USER_REQUIRED) for the existing LU.
- The application-based LU map does not match the already chosen LU.

TN3270 LU mapping: TN3270 connections only perform Complete Lookup after all information is known. LU lookup is not done during connection negotiation. Telnet will either send a solicitor (or USSMSG10) screen to the client or will perform Complete Lookup using the application name known through the LUMAP-DEFAPPL or DEFAULTAPPL statement. If Complete Lookup is successful, Telnet will begin session initiation. If not successful, the solicitor (or USSMSG07) screen is sent to the client without an LU being assigned to the connection or the connection is dropped. The LU is not assigned until the application name is valid. If the application name is a RESTRICTAPPL, the LU is not assigned until a user ID is specified. Application-based LU mappings have a very good chance of success due to the late LU mapping aspect of TN3270 connections. When SIMCLIENTLU is coded, Generic TN3270E connections have this same characteristic.

Complete Lookup: An application name is required for Complete Lookup. The application name is obtained from one of three sources in the order specified.

1. Input from the USER or VTAM (CLSDST with OPTCD=PASS)
2. DEFAPPL parameter on the LUMAP statement
3. DEFAULTAPPL statement

Use the application name to perform Complete Lookup. Possible lookup results are:

- The application is not valid.
- The application is valid but an LU is not found.
- The application is valid (return code OK) and an LU is found.
- The application LU map does not match the Client Identifier LU map.

If the application is not valid, no LU is assigned to the connection and an error message is sent to the client. If the application is valid, continue the LU lookup in the following order.

- Check for LUMAP matches considering application-based LU lookup results. Only Generic LUMAPs are searched. If the application lookup return code is OK, then perform LU lookup.
- If no LUMAP statements were used, check for application-based LU mappings. If the application lookup return code is OK and LUs are defined on the application statement, perform LU lookup.
- If no LUMAP or application-based LU mapping statements were used, use the DEFAULTLUS pool considering application lookup results. If the application lookup return code is OK, then perform LU lookup.

Advanced application topics

In addition to the basic function of facilitating session setup, Telnet supports several advanced functions such as:

- Session initiation management (LOGAPPL, QINIT, FIRSTONLY, and DEFONLY)
- Connection and session takeover
- Queuing sessions
- Disconnect on session error
- Bypass RESTRICTAPPL with CERTAUTH
- Keeping the ACB open
- Express Logon Feature

Session initiation management (LOGAPPL, QINIT, FIRSTONLY, and DEFONLY)

The LOGAPPL, QINIT, FIRSTONLY, and DEFONLY options can be coded on DEFAULTAPPL, PRTDEFAULTAPPL, LINEMODEAPPL, LUMAP-DEFAPPL, or PRTMAP-DEFAPPL. For the remainder of this section, DEFAULTAPPL represents all the default application statements.

LOGAPPL, QINIT: The LOGAPPL or QINIT functions keep the Telnet LU active if a Request Session fails due to the host VTAM application not being active. In addition, VTAM remembers the attempted Request Session and will initiate a session request to the Telnet LU on behalf of the application when the application becomes active. When the Request Session fails, Telnet sends the client a solicitor panel or USSMSG07 screen. The end user then has the option of logging on to a different host VTAM application (if DEFONLY is not coded). When this different session is started, VTAM drops the queued Request Session for the original session.

What happens at session logoff depends on whether or not LUSESSIONPEND and FIRSTONLY are coded and whether LOGAPPL or QINIT is coded. If LUSESSIONPEND is coded, the connection remains. Otherwise, it is dropped. If FIRSTONLY is coded, Telnet will send a USSMSG10 screen or Solicitor panel to the client. If FIRSTONLY is not coded, Telnet will initiate another session with the

default application defined by the DEFAULTAPPL statement. LOGAPPL and QINIT have different results when logging off the original application. When LOGAPPL is specified, a USSMSG10 or Solicitor panel is sent to the client. When QINIT is specified, the default application is redriven.

FIRSTONLY, DEFONLY: Sometimes a default application is used at initial connection, but after LOGOFF a USSMSG10 or solicitor panel is more appropriate than redriving the default. In this case, code the FIRSTONLY parameter. This indicates the default should be used on the first session only. After a session has been established, any subsequent lookups will ignore the default and send the USSMSG10 screen or solicitor panel.

If MSG07, LUMAP-DEFAPPL, or PRTMAP-DEFAPPL is coded and the default application is inactive, an error screen will be sent to the client. The DEFONLY parameter will block a user-entered application choice if it is different than the default. This parameter prevents application choice while giving the end user error information.

The following table summarizes several possible session initiation failure scenarios.

- ReqSess OK - A Request Session to the default application succeeded or a Request Session to a second application from the USSMSG07 screen succeeded.
- ReqSess Fail - A Request Session to the default application failed.
- 2nd Appl Fail - A Request Session to a second application from the USSMSG07 screen failed.

Scenario / Mapping Statement	ReqSess OK	ReqSess Fail		2nd Appl Fail	
		MSG07	No MSG07	MSG07	No MSG07
DEFAULTAPPL name	1	2	5	2	N/A
DEFAULTAPPL name LOGAPPL DEFAULTAPPL name QINIT	1	3	3	4	4

Figure 55. Session initiation failures scenarios

FIRSTONLY is not a consideration because the first session has not been established.

1. In session.
2. Send USSMSG07 or Solicitor panel to client. Close the ACB.
3. Send USSMSG07 or Solicitor panel to client. Keep ACB open, queue original session request in VTAM.
4. Send USSMSG07 or Solicitor panel to client. Keep ACB open, keep the new queued session request in VTAM.
5. Drop connection.

The following table summarizes several possible session ending scenarios. The session is ending due to normal LOGOFF or session breakage (possibly caused by loss of the application).

- Original Application - User is in session with the original default application.
- CLSDST from Original Appl - User is in session with a second (or later) application after issuing a CLSDST-PASS from the original application.

Scenario / Mapping Statement	Original Application		2nd Appl or CLSDST from Orig	
	Logoff	Break	Logoff	Break
DEFAULTAPPL name DEFAULTAPPL name QINIT	1	1	1	1
DEFAULTAPPL name LOGAPPL	2	1	1	1
DEFAULTAPPL name FIRSTONLY LOGAPPL DEFAULTAPPL name FIRSTONLY QINIT	2	1	2	1

Figure 56. Session ending scenarios

In all cases, if LUSESSIONPEND is not coded the connection is dropped.

1. Redrive the default application.
2. Send USSMSG10 or Solicitor panel to client. Close the ACB.

Connection and session takeover

In some cases the route for a connection is lost without Telnet being notified. When this occurs, the end user can no longer make contact with the host application. The end user will typically disconnect the emulator and try to start another session. In many cases this does not work. For example, assume the host application is TSO and the end user is in session with TSO user ID USER1. The route is lost, so the end user disconnects and establishes a new connection over a different route. Assume a Generic connection so Telnet will assign a different LU to represent the client. When the TSO logon to USER1 is attempted, TSO will fail the logon because USER1 is still in session with the original Telnet LU. There is no way for the end user to bring down the original connection. The end user has to wait for an inactivity timer in Telnet or TSO to bring down the original connection. If a Specific connection is requested the connection is rejected sooner. The second Specific connection request will specify the original LU. Telnet will fail the request during Early Lookup indicating that the LU is already in use. The problem with both situations is that the original connection is still assumed active by Telnet.

The TKOSPECLU parameter fixes this problem if the end user knows the Telnet LU name of the original connection. The statement name is derived from the function of connection takeover (TKO) using the Specific LU (SPECLU) name. The Specific request is required as a security measure and it is assumed that most users of this function will have first connected using a Specific request. When the connection request arrives, Telnet discovers the LU name is in use and suspends the new request. Telnet sends a TIMEMARK request to the original client which acts as an ″are you there″ message. The client is required to respond to the TIMEMARK. If no response is received by Telnet within the time specified on the TKOSPECLU statement, Telnet drops the original session and connection. The original LU is reserved during the drop process. Once the original session and connection are dropped, Telnet resumes processing the new request. This time the LU is not in use, only reserved for takeover purposes, and is assigned to the new takeover session. The end user is essentially starting over. The original session has been dropped allowing the end user to immediately log on to the same TSO user ID again.

The TKOSPECLURECON parameter can be used to accomplish the same connection drop but avoids the session drop. When the original connection is dropped, the Telnet LU stays in session with the host application. The new connection is established and Telnet sends an LUSTAT to the host application

indicating that Presentation Space Integrity was lost ('082B'x). Depending on the application, it will either end the session or resend the previous screen. By resending the previous screen, the end user is able save the original session and avoid the SNA session tear-down and restart process. At worst, if the application drops the session upon receipt of the LUSTAT the end user is able to immediately log on again as if TKOSPECLU were coded.

TKOSPECLU and TKOSPECLURECON can be coded at all three parameter block levels for different levels of granularity. Code NOTKO to turn off the function at any of the three levels.

If the original connection used SSL with CLIENTAUTH SAFCERT, either takeover method will verify that the new connection is using the same client certificate. Telnet does this by translating the new certificate to a user ID and comparing the new user ID to the user ID on the original connection.

CAUTION: TKOSPECLURECON does not require the end user to reverify user authenticity to the host application. If there is a chance the connection can be taken over by an unauthorized user, TKOSPECLURECON should NOT be used.

CAUTION: A time value of zero is permitted. In this case the server will always perform the takeover whether or not the original connection is still active. The zero value is intended for testing purposes rather than production use.

Sometimes a takeover attempt will not complete as expected. This may be due to one of the following factors:

- The profile of the original connection defines how the original connection can be taken over. Be sure the original connection supports the desired takeover method.
- The new connection request must specify the LU name of the connection being taken over.
- TKOSPECLURECON will not preserve a session if the takeover is done from a different port. The ACB of the LU is associated with the original port and must be closed before it can be associated with the new port. The takeover will function as a TKOSPECLU takeover.
- TKOSPECLURECON may not preserve a session if the original client connection is ended before the TKOSPECLURECON timer expires. If the close reason is TIMEMARK or INACTIVE, the session will be preserved under the assumption that the inactivity is due to a lost connection. Any other close reason will cause the takeover to function as a TKOSPECLU takeover. This is done to protect users who disconnect their client as a means of logging off their session. These sessions will not be taken over. Instead, the end user will have to issue a new logon. In some cases, the original client TCP/IP stack may respond to the Timemark with a RESET. Telnet interprets this RESET as a client disconnect and assumes the end user disconnected the session. Telnet then drops the session. To keep the session in this case, add the KeepOnTmReset option to the TKOSPECLURECON parameter. A security risk exists when using this parameter. An end user may actually disconnect just before the Timemark arrives due to an unauthorized takeover attempt. Telnet will interpret the disconnect as a response to the Timemark and allow the takeover without loss of the VTAM session.

Takeover is also affected by where the new client resides and how the old client responds. There are several event scenarios and results will vary.

- Event 1

New client connects from a different IP address or Port.

Original client responds to TIMEMARK.

Result - Takeover will not occur in this case because the original client is still responding. The new client will receive an error indicating the LU is already in use.

- Event 2

  New client connects from a different IP address or Port.

  Original client does not respond to TIMEMARK.

  Result - Connection takeover will occur. If TKOSPECLURECON is mapped to the client, session takeover will occur. A likely scenario in this case is Telnet has lost connectivity to the old connection due to a failed router and the new connection is using a different route, the original machine lost power and has not reestablished connectivity, or the original machine lost power but reestablished connectivity with a different IP address.

- Event 3

  New client connects from a different IP address or Port.

  Original client stack responds with RESET.

  Result - Connection takeover will occur. However, even if TKOSPECLURECON is coded, session takeover will not occur because Telnet handles the RESET as a client disconnect. A likely scenario in this case is a PC lost power and then regained power. The takeover request is accepted from either a different PC or the same PC using a different port. After the new connection request is accepted, Telnet sends a TIMEMARK to the original client PC stack. The PC stack does not recognize the IP-port and responds with a RESET. If KeepOnTmReset is specified and the RESET is received by Telnet after the Timemark has been sent, Telnet will keep the session.

- Event 4

  New client connects from the same IP address and Port.

  Telnet stack rejects the request.

  Original client times out and sends a RESET.

  Result - The original session and connection are dropped. Takeover does not occur. The new client is able to connect on retry because the original connection and session were cleaned up. A likely scenario in this case is a PC has lost power and then regained power. The same PC is used to attempt takeover. The client is assigned the same port as before the power loss and has the same IP address.

Queuing sessions

Logon manager applications are very popular. Typically they are set up as a default application which sends a selection screen to the end user. Once the end user specifies the destination application choice, the logon manager typically issues a CLSDST macro with OPTCD=PASS to the destination application. A new session is started with the destination application. The logon manager session is closed with a special UNBIND sent to Telnet indicating that a new session BIND is forthcoming. Telnet receives that special UNBIND and then waits for the next BIND instead of cleaning up as it would when receiving a normal UNBIND. When the end user logs off the destination application, Telnet will either redrive the initial database lookup process or drop the connection, depending on whether or not LUSESSIONPEND is coded.

Many logon managers were written to support real terminals, not Telnet, and have the added function of issuing a SIMLOGON Q for the logon manager session itself

immediately after issuing the CLSDST-PASS. When SIMLOGON is coded, VTAM (on behalf of the logon manager) will request a session with the terminal LU (or Telnet LU representing the client) immediately after user logoff from the destination application. This works very well for real terminals, but causes timing problems for Telnet when the logon manager is a default application. In this case, Telnet and VTAM both end up requesting a session.

The QSESSION option on ALLOWAPPL or RESTRICTAPPL can be used to correct this timing problem. When coded for the logon manager, Telnet will *not* do normal close processing when the UNBIND from the destination application arrives. Telnet will leave the LU ACB open and wait for the BIND from the logon manager that is generated because of the Queued SIMLOGON. When the BIND does arrive, Telnet will verify that the application name is the original logon manager and finish session setup.

You must ensure that the application defined as the QSESSION application will issue a SimlogonQ so a BIND will be sent after logoff from the destination application. Otherwise, Telnet will leave the LU ACB open waiting for a BIND that is never coming. The connection will be essentially hung. Only a Telnet expiration timer or client disconnect will clean up the connection. Logoff of the original application will cause Telnet to perform normal close function instead of leaving the LU ACB open.

The BEGINVTAM statement, QUEUESESSION, provides the same function with less control than the QSESSION option and will be made obsolete in a future release. QUEUESESSION affects all defined DEFAULTAPPLs whether or not all are queue session applications. QUEUESESSION precludes the use of LUSESSIONPEND and QSESSION does not. If you are using QUEUESESSION, you should update your profile to use the QSESSION option instead.

As an example of the QSESSION parameter, assume that APPL1, APPL2, and APPL3 are each defined in VTAM. APPL1 will issue a SimlogonQ before CLSDST-PASS. The following Telnet statements allow connections to access the applications and define which is a QSESSION application.

```
ALLOWAPPL APPL1 QSESSION
ALLOWAPPL APPL*
```

The client first logs on to APPL1 and Telnet saves the name in the first slot of a 10 slot Qsession table. A CLSDST-PASS to APPL2 and a SimlogonQ are issued. The APPL2 name is saved in slot two of the table. Finally, a CLSDST-PASS to APPL3 is done and the APPL3 name is saved in slot three of the table. When the APPL3 session is ended, the APPL3 slot is cleared and VTAM sends an APPL1 BIND to Telnet. The first application queued is the first application off the VTAM queue. In this case Telnet will start at the end of the Qsession table (APPL2 since APPL3 entry was just cleared) and check each slot to try to find an application name match. Eventually a name match is made with slot one. Now, all slots above slot one are cleared. A CLSDST-PASS to APPL4 will cause the APPL4 name to be put into slot 2. Because the slots for APPL2 and APPL3 were cleared earlier, Telnet will no longer accept a BIND from APPL2 or APPL3 after ending the session with APPL4.

The Qsession table is set up when a BIND is received for an application that is defined with the QSESSION parameter. If the first application does not have QSESSION coded and a CLSDST-PASS is issued to an application name that has QSESSION coded, it is the second application that will be in slot one of the

Qsession table. When that session in slot one is ended, Telnet will clean up the LU and the connection status will depend on whether or not LUSESSIONPEND is coded.

Disconnect on session error

The DISCONNECTABLE option on either the ALLOWAPPL or RESTRICTAPPL statement determines what type of session termination to send to the host VTAM application when Telnet initiates session termination. If DISCONNECTABLE is coded, Telnet issues a TERMSESS UNBIND(0F). Otherwise, Telnet issues a TERMSESS UNCOND. For example, when DISCONNECTABLE is coded for the TSO application, an unexpected connection loss results in an UNBIND(0F) being sent to TSO putting it in a reconnectable state. The DISCONNECTABLE parameter has no effect on a session ended normally by the end user logging off the session. The QSESSION parameter can be coded with DISCONNECTABLE on either statement.

Bypass RESTRICTAPPL with CERTAUTH

CERTAUTH is an option on RESTRICTAPPL used in conjunction with client authenticated secure connections or Express Logon. In both cases the client certificate is used to derive a user ID. If the end user chooses an application that is a RESTRICTAPPL, the normal Telnet response is to request a valid user ID and password before allowing access to the application. However, if the end user has been authenticated with a client certificate it may not be necessary to require a user ID/password. With the CERTAUTH option on RESTRICTAPPL Telnet will use the derived user ID. If the user ID is valid (listed on the RESTRICTAPPL statement), Telnet will bypass the end user solicitation and immediately give access to the application. The derived user ID value depends on the type of connection. If Express Logon is being used, the user ID is derived from the latest Client Certificate/Applid combination received from the client. If Express Logon is not being used, the user ID is the Client Identifier user ID derived from the Client Certificate after the SSL handshake.

Keeping the ACB open

Some host VTAM applications are set up to inquire if a secondary LU is active. If the LU is active, the application will initiate a session. For this to work, the secondary LU must be active. The LUMAP option, KEEPOPEN, will cause the ACB to open when the LU is assigned to the connection and remain open for as long as the LU is assigned to the client by this mapping statement.

LU assignment is different for TN3270E and TN3270 connections. TN3270E connections have an LU assigned during connection negotiation. TN3270 connections do not have an LU assigned until the VTAM application name is known.

In some cases the host VTAM application will initiate a session with the secondary LU. To do this, the host must issue an INQUIRE to see if the LU is active with an OPEN ACB. This INQUIRE will fail for Telnet LUs because Telnet does not open the ACB until a session request is sent from Telnet to VTAM. When the end user gets the solicitor (or USSMSG10) panel, Telnet has not opened the ACB of the LU assigned to a TN3270E connection. TN3270 and LineMode connections do not have LUs assigned yet. If the KEEPOPEN parameter is coded on the LUMAP statement used by Telnet to assign an LU during Early Lookup for a terminal TN3270E connection, Telnet will open the ACB before sending the solicitor (or USSMSG10) panel. At that time an end user can either log on to an application as usual or wait for a host application to INQUIRE about the LU and initiate a session. TN3270 connections will not have an LU assigned until an application name is known. When the name is known, an LU is assigned to the connection, the ACB of

the LU is opened, and a session request is issued. If the KEEPOPEN parameter is coded on the LUMAP statement used by Telnet to assign the LU, the LU stays assigned to the connection with the ACB open until the connection is dropped. If profile statements define LUs uniquely to different applications, a second logon to a different application might fail. When KEEPOPEN is mapped to a connection, the MSG07 and LUSESSIONPEND functions are in effect whether or not they were explicitly coded. When a session is ended, the connection remains and the ACB remains open. Only a client disconnect, a Telnet error, or the KEEPINACTIVE/INACTIVE timers will cause a KEEPOPEN connection to be dropped. The KEEPINACTIVE timer is used whenever the Telnet LU is not in session with a VTAM application. Otherwise, the INACTIVE timer is used. See "Timers" on page 366 for information about ending idle KEEPOPEN connections.

Express Logon Feature (ELF)

The Express Logon Feature allows an end user to connect to an MVS host VTAM application without explicitly entering a user ID or password. Telnet uses the client certificate to resolve the user ID and RACF generates a temporary password, a passticket. ELF requires a secure connection with level 2 client authentication, a client that supports ELF, and RACF passticket setup.

The ELF function is activated by specifying the EXPRESSLOGON parameter. The function can be inactivated by specifying NOEXPRESSLOGON. Either parameter can be coded in TELNETGLOBALS, TELNETPARMS or PARMSGROUP. For a detailed discussion of ELF, refer to Appendix C, "Express Logon Feature (ELF)" on page 749.

Device types and logmode considerations

The VTAM logmode defines many characteristics of the session established between the Telnet LU representing the client and the host VTAM application. For example, the logmode defines response types, presentation style, and the type of LU Telnet is emulating. LU0 (non-SNA) and LU2 (SNA) represent terminal LU types. LU1 (SCS) and LU3 (3270 Data) represent printer LU types.

Telnet matches a VTAM logmode to each client as it connects based on the client device type. Refer to *z/OS Communications Server: IP Configuration Reference* for default device type and logmode table information. The default terminal logmodes are non-SNA for TN3270 connections and SNA for TN3270E connections. At session request time, Telnet indicates to VTAM the desired logmode based on device type. The host VTAM application usually honors the request and binds the session using the requested logmode. However, depending on VTAM statements, the application can override the requested logmode and bind the session using different characteristics than Telnet requested. For this reason, some screen sizes might not work correctly even though the logmode defined in Telnet is correct. If the KEEPOPEN function is used to allow session initiation by the host application, the desired logmode must be coded on the DLOGMOD parameter as part of the VTAM application definition statement that defines the Telnet LU. Otherwise, the application will choose its own logmode.

Telnet processes the ATTN KEY request differently for non-SNA and SNA sessions. For non-SNA sessions (BIND FM value 02), Telnet converts the ATTN KEY request to a '6C'x data byte and sends it to the application. For SNA sessions (BIND FM value 03), Telnet converts the ATTN KEY request into a SNA signal and sends it to the application as expedited data. Some clients send both an ATTN KEY function code and a '6C'x data byte to ensure the ATTN is seen by the application. Telnet converts the ATTN KEY function into either a '6C'x data byte or a SNA signal and also forwards the '6C'x data. Some applications give unexpected results or Telnet

might appear to not support ATTN when two ATTNs are received. The SINGLEATTN parameter causes Telnet to drop the second ATTN if it immediately follows an ATTN. The SINGLEATTN and NOSINGLEATTN parameters can be coded at all three parameter block levels for different levels of granularity.

To change either the TN3270 or the TN3270E logmode for a device type, use the TELNETDEVICE parameter. Whenever Telnet initiates the session request, Telnet will request that the logmode specified on the TELNETDEVICE statement be used for the session. The application (the primary LU) does have the ability to override the requested logmode and use a completely different logmode. The TELNETDEVICE parameter can be coded in all three parameter block levels for different levels of granularity. Coding TELNETDEVICE in a PARMSGROUP that is mapped on the LUMAP-DEFAPPL-PMAP statement enables the logmode to be LU and application specific.

If the application initiates the session, the TELNETDEVICE logmode has no affect on the session. For example, printer sessions are initiated by the application unless a printer default application is specified. The LUMAP-KEEPOPEN parameter can be used to open a terminal ACB and wait for the primary application to initiate the session.

Transform Linemode connections can have a unique logmode by coding TELNETDEVICE with a device type of TRANSFORM. Any logmode used must not support extended graphics.

The special case logmode NONE can be specified indicating that Telnet should not send any logmode request when initiating the session.

In the examples that follow, the first line causes only the 3270 logmode to change from the default to SNX32705. The second line causes both the 3270 and 3270E logmodes to change from their defaults to SNX32705 and SNX32702. The third line causes only the 3270E logmode to change from the default to SNX32702.

```
TELNETDEVICE 3278-5-E SNX32705
TELNETDEVICE 3278-5-E SNX32705,SNX32702
TELNETDEVICE 3278-5-E ,SNX32702
```

Using the Telnet Solicitor or USS logon panel

This section describes the Telnet Solicitor panel and Telnet Unformatted System Services (USS) support. All information needed to establish a session can be entered on the Telnet Solicitor panel. However, Telnet is often used as the primary method of connecting to the SNA mainframe environment. SNA end users are accustomed to entering abbreviated logon commands, altering logmodes, and entering user data from SNA terminals. For ease of migration, Telnet simulates SNA USS processing very closely. This simulation extends to being able to use the same assembled USS tables that are used by VTAM. VTAM-only character substitutions are ignored by Telnet and Telnet-only character substitutions are ignored by VTAM. Blanks are used in their place. To further extend the simulation of SNA terminals, Telnet also supports all of the INTERPRET table function.

Using the Telnet Solicitor logon panel

Telnet sends a Solicitor panel to the end user if one of the following is true:

- No DEFAULTAPPL, LINEMODEAPPL, USSTCP, or LUMAP-DEFAPPL mappings match the client's Client Identifier.
- The requested application is a RESTRICTAPPL.

Below is an example of the Telnet Solicitor panel:

```
Enter Your Userid:
Password:                          New Password:
Application:
```

Initial cursor placement can be specified. Where initial placement should be depends on client macros used and end user preferences. The OLDSOLICITOR parameter is used to implement this choice. The default cursor position is on the 'Application:' field. If OLDSOLICITOR is coded, the cursor is positioned on the 'Enter Your Userid:' field. The OLDSOLICITOR and NOOLDSOLICITOR parameters can be coded at all three parameter block levels for different levels of granularity.

In addition to satisfying RESTRICTAPPL, there are other times when an end user might want to use the user ID/Password fields. For example, the solicitor panel may also be used to change a password by entering the user ID, old password, and new password. The application field does not need to be filled in. If insufficient information was provided by the client (for example, a user ID but no password), then the Telnet Solicitor panel is returned with a message prompting for the required field. A message is also returned if the security program encounters an error when attempting to change the password. Example messages include:

- Password required
- Password is not authorized

Using the Telnet USS and INTERPRET support

The Telnet USS function provides the end user with a USSMSG10 logon panel similar to the logon panel used by native SNA terminals. The Telnet USS function supports sending USSMSGs to the client, receiving and parsing USSCMDs from the client, and using a translation table defined in the USS table.

For any USSMSG other than USSMSG10, pressing the CLEAR key will refresh the screen with USSMSG10. If the CLEAR key is pressed while at USSMSG10, the screen will be cleared. Pressing the CLEAR key a second time will refresh the screen with USSMSG10.

USSCMD parsing also includes checking for INTERPRET table entries that might provide more function than USS tables alone can provide. Sample USS and INTERPRET tables are in TCP/IP data sets hlq.SEZAINST(EZBTPUST) and hlq.SEZAINST(EZBTPINT), respectively. The USS sample has been assembled, linked, and loaded into the product data set. The tables can be used by coding the USSTCP and INTERPTCP mapping statements in BEGINVTAM. For example, the statements below will map the sample tables to the client at IP address 1.1.1.1. See "Mapping Objects to Client Identifiers" on page 325 for mapping details.

```
USSTCP   EZBTPUST  1.1.1.1
INTERPTCP   EZBTPINT  1.1.1.1
```

A new table can be created at any time and link-edited. Customized USS and INTERPRET tables can be created to change messages, commands, and translation tables. For example, messages can be changed to have non-English text or to have different syntax. Commands can be changed to accept different syntax or to have different default values. A VARY TCPIP,,OBEYFILE command will cause Telnet to load the new table with the new profile being processed. Any new connection using the new profile will be assigned the new table. Telnet also supports dynamic updating of same-name USS or INTERPRET tables. VARY TCPIP,,OBEYFILE adds the new version of the table to the new profile. New connections use the new copy associated with the new profile while old connections continue to use the old copy associated with the old profile.

USS table customization: Customized USS tables are used by both VTAM and Telnet, with any product-specific character substitutions converted to blanks. For example, @@SSCPNM is blank for Telnet and @@PRT is blank for VTAM. The tables must reside in a data set that is in the system's linklist or is in the STEPLIB statement of the TCP/IP startup procedure. Any changes to a Telnet USS table should be made with supplementary user-defined USS tables. The IBM-supplied USS table should not be changed as it provides a good example of coding most commands and messages. Telnet loads the first table found with the name EZBTPUST and defines it as the default USS table. If this table is not found, there is no default USS table. Whether or not a default USS table should be included depends on the desired message output. When writing a USS Message, Telnet searches the USS table mapped to the client first. If the message does not exist in the mapped table, Telnet searches the default table. If the message does not exist in the default table, Telnet writes USSMSG14. If no default table exists, Telnet generates a USSMSG14. The end user can get back to the USSMSG10 from any message by pressing the CLEAR key. The default table does not affect the USS Commands. The command entered must be in the mapped table or it is not recognized.

Creating a USS table: The following macro instructions are used to create the USS table. Telnet USS function supports almost all VTAM session-level USS message and command definitions. Refer to *z/OS Communications Server: IP Configuration Reference* for macro details.

- USSTAB indicates the beginning of the USS table.
- USSCMD defines commands accepted by the Telnet server.
- USSPARM defines each operand or positional parameter that can be specified on the USSCMD macro instruction. It also defines default values for the operand or positional parameter. Multiple USSPARM macro instructions can be associated with a USSCMD macro instruction. For each operand or positional parameter code a USSPARM macro instruction.
- USSMSG defines messages sent from the Telnet server.
- USSEND indicates the end of the USS table.

Below are some of the more common rules to consider when coding a new USS table. Also, refer to the sample found in hlq.SEZAINST(EZBTPUST) as a guide. The following section discusses general table rules.

- If a DEFAULTAPPL application is mapped at the same Client Identifier level as a USS table, the USS table will only be used to return error messages and optionally after the first session logoff. An LUMAP-DEFAPPL application will always be used instead of the USSMSG10 regardless of Client Identifier priority. FIRSTONLY or LOGAPPL options on DEFAULTAPPL will cause Telnet to send a USSMSG10 after the first session logoff. DEFAULTAPPL without the FIRSTONLY or LOGAPPL options will cause Telnet to redrive to the default application after every session logoff.
- Only the 3270 data stream is supported. Refer to *3270 Data Stream Programmer's Reference* for more information.
- If a user-defined table is coded as part of another module, code an assembler EXTRN definition statement for the table name in that module so the table will be known externally and can be accessed by other modules.

Below are message related rules.

- USSMSGs must contain the 3270 data stream write control characters (WCCs).
- All character substitutions (@@'s) substitute the same number of fields. Any character substitution that is VTAM-specific will be translated to blanks. If the

substituted value is smaller, the field is padded to the right with blanks. The parameter LUNAME or SCAN must be coded on the USSMSG macro instruction for Telnet to perform character substitutions. For a complete list of character substitutions, refer to *z/OS Communications Server: IP Configuration Reference* for Telnet and *z/OS Communications Server: SNA Resource Definition Reference* for VTAM. Telnet supports multiple USSPARMs with the DATA keyword. This method can be used to pass multiple data parameters to the host application. For example, two DATA USSPARMs allow the end user to type 'TSO USER1 PROC001' and have both the user ID and the Procname passed to TSO as data.

Below are command related rules.

- LOGON command format

  PL1 - logon applid(tso) logmode(snx32702) data(user1)

  BAL - logon applid=tso,logmode=snx32702,data=user1
- Any application defined in a USSCMD macro instruction must also be specified on either an ALLOWAPPL or a RESTRICTAPPL statement in the Telnet Profile.
- If the USS Command rules in *z/OS Communications Server: IP Configuration Reference* cannot be followed, use an interpret table to convert the character-coded command into a formatted SNA request.

INTERPRET table customization: The standard Telnet USS logon support should meet the needs of most installations. However, Telnet does support INTERPRET table function if special circumstances require accepting a sequence of characters outside the normal USS command format. For example, the end user might want to enter logon data that includes blanks. The INTERPRET table defines all entered data, including blanks, as a USSPARM DATA entry. The PL1 USSCMD format treats each blank as a parameter delimiter and cannot properly process a variable number of parameters. The INTERPRET table character sequences are scanned whenever the client is mapped to both a USS table and an INTERPRET table. Both must be mapped because the INTERPRET function is a subset of the USS function. INTERPRET is not a stand-alone function. The sample INTERPRET table found in hlq.SEZAINST(EZBTPINT) is not assembled and linked, and it is not loaded into Telnet as a default INTERPRET table. The table must be assembled, linked, loaded, and mapped in the Telnet profile to be used.

Creating an INTERPRET table: Telnet INTERPRET function supports all functions provided by the VTAM INTERPRET definitions. Refer to *z/OS Communications Server: IP Configuration Reference* for macro details. The following macro instructions are used to create an INTERPRET table:

- INTTAB indicates the beginning of the INTERPRET table.
- LOGCHAR defines a single logon message and name of an application program.
- ENDINTAB indicates the end of the INTERPRET table.

Below are some of the more common rules to consider when coding a new INTERPRET table. Also, refer to the sample found in hlq.SEZAINST(EZBTPINT) as a guide.

- The LOGCHAR APPLID= supports APPLICID, ROUTINE and USERVAR.
- Code the most restrictive, or longest, LOGCHAR SEQNCE values first. Otherwise, unexpected matches can occur. The table is scanned from top to bottom until a match is found whether or not it is the most exact match. For example, assume sequence 'LOGA' is assigned APPL1 and any other 'LOG' sequence is assigned APPL2. If sequence 'LOG' is before 'LOGA', entry 'LOGA'

will never be found even when the end user enters 'LOGA' because entry 'LOG' will be the first match. All sessions will go to APPL2. The problem is corrected by putting 'LOGA' before 'LOG' in the table.

Assemble, link, and load a table: Use the sample JCL in hlq.SEZAINST(EZBUSJCL). In the sample, the USS table is in USER1.TABLES(USSTEST). It must be assembled and link-edited into the system's linklist or into a library concatenated as a STEPLIB in the TCP/IP startup procedure. In the sample, the table is link-edited into USER1.LINKLIB(USSTEST). The same procedure can be used for the INTERPRET table. Simply change the name of the input file source and the link-edit target member. The VTAM USS and INTERPRET macros used for the assemble can be found in hlq.SISTMAC1.

Timers

Several timers are available in Telnet to control how long connections stay up. The list includes:

- INACTIVE - How long a terminal connection can be idle with no SNA data traffic before the connection is dropped.
- PRTINACTIVE - How long a printer connection can be idle with no SNA data traffic before the connection is dropped.
- KEEPINACTIVE - How long a KEEPOPEN connection can be idle with no SNA session before the connection is dropped. When a KEEPOPEN connection is in session with a SNA application the INACTIVE timer is used instead of the KEEPINACTIVE timer.
- SCANINTERVAL - How often Telnet runs the list of connections looking for potentially lost connections. Because of the methodology, it also determines how long Telnet will wait for a TIMEMARK response before assuming the connection is lost.
- TIMEMARK - How long a connection is active without receiving any data before Telnet sends a TIMEMARK command which acts as an ″are you there″.
- SSLTIMEOUT - How long Telnet will wait for an SSL handshake initiation from the client before the request is dropped.

To facilitate these timers, Telnet records the time at which data is received from the client, received from VTAM, or sent to VTAM. Data received from the client is used by SCANINTERVAL/TIMEMARK to measure idle time on the connection. Data received from or sent to VTAM is used by the INACTIVE family of timers to measure idle time without SNA data traffic.

INACTIVE, PRTINACTIVE and KEEPINACTIVE all share one timer associated with a port profile to reduce system overhead. The timer with the smallest value defined in TELNETGLOBALS, TELNETPARMS or PARMSGROUP for that port profile is used to define how often the connections are checked. For example, assume KEEPINACTIVE is defined as 1800, INACTIVE is defined as 3000, and PRTINACTIVE is defined as 5400 in a profile. The Telnet timer will run every 1800 seconds. Therefore, every time the timer pops Telnet will check each KEEPOPEN connection not in session to see if there has been a SNA session created in the prior 1800 seconds. If not, the connection is dropped with DEBUG SUMMARY message CONN DROP reason INACT-K. Telnet will also check each terminal connection to see if there has been any SNA data traffic in the prior 3000 seconds. If not, the connection is dropped with DEBUG SUMMARY message CONN DROP reason INACT-S. Telnet also will check each printer connection to see if there has been any SNA data traffic in the prior 5400 seconds. If not, the connection is

dropped with DEBUG SUMMARY message CONN DROP reason INACT-P. Setting KEEPOPEN to the smallest time was done as an example. Any of the three timers could be the smallest.

SCANINTERVAL and TIMEMARK are used together to determine if a connection has been lost. These parameters can be specified in TELNETGLOBALS, TELNETPARMS, and PARMSGROUP. The smallest SCANINTERVAL value is used to define how often the connections are checked. If TIMEMARK is smaller than SCANINTERVAL, TIMEMARK is set equal to SCANINTERVAL. Whenever data is received from the client, Telnet records the time. Telnet checks all connections at regular intervals defined by the SCANINTERVAL value. Each connection is checked to see if any data has been received from the client in the past TIMEMARK period of time. If not, a TIMEMARK command is sent to the client which acts as an ″are you there″ and Telnet remembers a TIMEMARK was sent to this client. During the next check at SCANINTERVAL time later, each connection is again checked to see if any data has been received from the client. If not, and a TIMEMARK was sent on the previous check, the connection is dropped with DEBUG SUMMARY message CONN DROP reason TIMEMARK. For example, assume the values for SCANINTERVAL and TIMEMARK are 1800 and 10800, respectively. That means every 30 minutes all connections are checked to see if any data has been received in the last 3 hours. If not, a TIMEMARK is sent to the client. 30 minutes later Telnet checks the connections again. If the client responded to the TIMEMARK or sent in actual data of some type Telnet leaves the connection active. If nothing has been received Telnet drops the connection.

Caution must be used in setting these timers. Setting the INACTIVE family of timers or SCANINTERVAL timer too low could cause excessive CPU usage. Setting the TIMEMARK value too low could also cause excessive flooding of the network with TIMEMARK commands. For example, these timers should take into account extended breaks such as lunch. If TIMEMARK is smaller than the lunch break time, the network may be flooded with TIMEMARK commands around the lunch hour. Be aware of the default values and be sure to set appropriate values for the situation.

SSLTIMEOUT is different than the other timers. Telnet does not run this timer. The time value is passed to the SSL handshake process. If SSL does not get a response from the client within SSLTIMEOUT period of time, the handshake request fails. Telnet will then proceed to the next available connection negotiation method or drop the connection.

Telnet diagnostics

In addition to general diagnostic tools such as CTRACE and dumps which are described in *z/OS Communications Server: IP Diagnosis*, there are Telnet specific diagnostic tools available. Profile syntax can be checked, display requests can be issued, session initiation and termination can be tracked, and error messages can be specified.

DEBUG

The DEBUG SUMMARY statement provides tracking of connection status. A summary message is written when:

- A connection request is accepted by Telnet.
- Connection negotiation is complete.
- A session is established with the host application.
- A session is dropped.
- A connection is dropped.

LU name, Connection ID, and client IP address and port are included in each message. In the example below an end user connects to port 23. The connection is negotiated as a TN3270E connection and a session with TSO is established. The session is dropped because of client disconnect (CLNTDISC) and then the connection is dropped because of client disconnect.

```
EZZ6034I TELNET CONN 00000011 LU **N/A**  ACCEPTED      23
         IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 NEGOTIATED TN3270E
         IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 IN SESSION TSO0001
         IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 SESS DROP  CLNTDISC
         IPADDR..PORT 1.12.13.14..456
EZZ6034I TELNET CONN 00000011 LU TCPM1001 CONN DROP  CLNTDISC
         IPADDR..PORT 1.12.13.14..456
```

In addition to tracking major state changes and providing key information, the statements can be used for debug purposes. For example, an end user might be attempting a connection and something is not working. The ACCEPTED, NEGOTIATED, and IN SESSION messages are major connection milestones. Using the information provided and knowing whether or not these messages are displayed can provide many clues about the connection request. The SESS DROP and CONN DROP messages give a variety of reasons about why the drop occurred.

The DEBUG DETAIL statement may be needed if the DEBUG SUMMARY messages do not provide enough information to solve a problem. In addition to the summary messages listed above, DEBUG DETAIL will issue a message at the time of failure which displays the client IP address and port, connection ID, Telnet LU name, detecting module name, unique return code and brief explanation, and additional parameters if relevant. Some messages will be helpful to the system administrator and others will be helpful to IBM service. Refer to *z/OS Communications Server: IP Messages Volume 4 (EZZ-SNM)* message EZZ6035I for return code details.

In the example below the end user specified an application not in the Telnet profile and then disconnected at the client emulator.

```
EZZ6035I TELNET DEBUG CLIENT IPADDR..PORT 1.12.13.14..456
     CONN: 00000011  LU: TCPM1001  MOD: EZBTPGLU
     RCODE: 3012-00  Application name is invalid.
     PARM1: 00000000 PARM2: 00000000 PARM3: 00000000

EZZ6035I TELNET DEBUG CLIENT IPADDR..PORT 1.12.13.14..456
     CONN: 00000011  LU: TCPM1001  MOD: EZBTTRCV
     RCODE: 1001-02  Client disconnected from the connection.
     PARM1: FFFFFFFF PARM2: 00000461 PARM3: 00000000
```

The DEBUG EXCEPTION statement causes the CONN DROP message to be issued only for error conditions and inactivity reasons. DEBUG EXCEPTION is the default. If more than one connection was dropped for the same reason within 15 seconds, a single message with LU name MULTIPLE will be issued. For example, if MSG07 is not coded in the DEBUG DETAIL example above, the connection will be dropped after the lookup failure. The CONN DROP message will include the return code and indicate that an error caused the connection drop. The following message will be produced whether or not DEBUG was coded because of the error condition.

```
 EZZ6034I TELNET CONN 00000011 LU TCPM1001 CONN DROP  ERR 3012
          IPADDR..PORT 1.12.13.14..456          EZBTPGLU
```

The DEBUG TRACE statement generates messages containing data to and from the client, to and from VTAM, and to and from an LU name exit for a single connection. The TRACE option allows you to quickly see why a client is not connecting or why a session hangs. Once TRACE is added with VARY TCPIP,,OBEYFILE, the first client assigned tracing will be the only client traced. Another connection cannot be traced until the client currently being traced is dropped.

```
EZZ6034I TELNET CONN 00000080 LU **N/A** ACCEPTED   223
         IPADDR..PORT 9.14.6.42..36484
EZZ6035I TELNET DEBUG CLIENT IPADDR..PORT 9.14.6.42..36484
  CONN: 00000080  LU:           MOD: TO  CLNT
  <-C- FFFD28
  PARM1: 00000003 PARM2: 00000000 PARM3: 00000000
EZZ6035I TELNET DEBUG CLIENT IPADDR..PORT 9.14.6.42..36484
  CONN: 00000080  LU:           MOD: FRM CLNT
  -C-> FFFB28
  PARM1: 00000003 PARM2: 00000000 PARM3: 00000000
EZZ6035I TELNET DEBUG CLIENT IPADDR..PORT 9.14.6.42..36484
  CONN: 00000080  LU:           MOD: TO  CLNT
  <-C- FFFA2808 02FFF0
  PARM1: 00000007 PARM2: 00000000 PARM3: 00000000
```

The DEBUG OFF statement ensures that all debug messages are suppressed, including the exception CONN DROP messages.

Profile debug warning and error messages can be turned off by coding DEBUG OFF or DEBUG SUMMARY in the TELNETGLOBALS block and putting that block before any other Telnet blocks in the profile.

The DEBUG parameter may cause flooding of the operator's console. Console flooding concerns can be dealt with in several ways.

- DEBUG messages are, by default, assigned to routing code 11, the JOBLOG. The DEBUG option JOBLOG can be used for the same effect. However, the master console also receives routing code 11 messages by default. To stop the messages from going to the master console, issue VARY CN(01),DROUT=(11), which drops routing code 11 from the console. The other DEBUG option, CONSOLE, will direct the messages to the master console, routing code 2, and the teleprocessing console, routing code 8.
- If DEBUG messages are being used primarily for problem diagnosis, the VARY TCPIP,,OBEYFILE command can be used to keep the number of messages low. Bring up Telnet initially without DEBUG coded. When a problem appears, issue a VARY TCPIP,,OBEYFILE command for a Telnet profile that includes the DEBUG statement. Only new connections to the new profile will produce messages. Once data is obtained, issue another VARY TCPIP,,OBEYFILE command for a Telnet profile that omits the DEBUG statement.
- If the Client Identifier of the client having the problem is known, include DEBUG in a PARMSGROUP statement. Using PARMSMAP, map that group to the client. Debug messages for only that client will be issued.

The VARY TCPIP,,TELNET,DEBUG,OFF command can be issued to turn off DEBUG for all connections associated with all profiles, including the current profile. To turn on DEBUG again, issue a VARY TCPIP,,OBEYFILE command with the Debug option specified in the Telnet profile. Summary messages for CONN DROP due to errors or time-outs will also be suppressed. Use DEBUG EXCEPTION to retain these messages.

MSG07

The MSG07 parameter is very helpful when diagnosing problems. It allows Telnet to send a message to the client indicating an error occurred and what the error was. Something simple like a mistyped application name can be corrected by the end user without additional help. Even for more difficult problems, MSG07 provides a good starting point. It is recommended that MSG07 always be coded unless there are reasons not to send error messages to the client.

Abend trap

The VARY TCPIP,,TELNET,ABENDTRAP,*module,rcode,instance* command can be used to set up an abend based on the variables specified. Abendtrap has three variables:

module Is required. It is the module detecting the error. It can be wildcarded with asterisk (*) at the end. If a single * is used, any module reporting the specified return code will cause an abend. The module name ″OFF″ turns off an active trap.

rcode Is optional. It is the return code reported and cannot be wildcarded.

instance Is optional. It is the instance of the return code and cannot be wildcarded.

Below is an example setting the abend trap and then issuing a profile display to verify the trap is set. In the example, when EZBTTRCV reports an error code of 1001, Telnet will issue an abend with reason code ’3133’x. The state of the trap changes from ″ACTIVE″ to ″TRIPPED″. No more abends will be issued. Once tripped, the abendtrap command must be issued again to activate the trap. While the trap is active, the abend trap can be turned off by specifying ″OFF″ as the module name. An active trap cannot be changed directly. The current trap must be tripped or turned off before a new command is accepted.

```
V TCPIP,TCPCS6,T,ABENDTRAP,EZBTTRCV,1001
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPCS6,T,ABENDTRAP,EZBTTRCV,1001
EZZ6013I TELNET COMMAND ABENDTRAP EZBTTRCV COMPLETE

D TCPIP,TCPCS6,T,PROF
EZZ6060I TELNET PROFILE DISPLAY
 PERSIS  FUNCTION   DIA  SECURE   TIMERS  SMF   MAX  LINE
(LMTQAK)(OATSQSWHT)(DRF)(PCKLECX)(IKPSTS)(ITIT)(RSQ)(BSCTT)
 ------  ---------  ---  -------  ------  ----  ---  -----
 LMR*A*  ***SQ**HT  TJF  SSH*ESX  ***STS  ITIT  RSQ  *SC**
----- PORT:    23  ACTIVE           PROF: CURR CONNS:      0
------------------------------------------------------------
   ABENDTRAP       : EZBTTRCV  1001  FF  ACTIVE
4 OF 4 RECORDS DISPLAYED
```

SMF

SMF records are written when an end user establishes a session (SMF LOGN or TN3270 Server SNA Session Initiation record) and when the session is ended (SMF LOGF or TN3270 Server SNA Session Termination record). Optional SMF recording is controlled by using the SMFINIT and SMFTERM statements. Two different record formats are available: SMF type 118 and 119. The type 119 records were first introduced in z/OS V1R2 Communications Server, and are controlled by use of the TYPE119/NOTYPE119 operands on the SMFINIT and SMFTERM statements. The subtypes cannot be changed for type 119 records and are set to the STD values. The use of the STD operand or the specification of a nonstandard subtype number on the SMFINIT and SMFTERM parameters control the usage of the older type 118 record processing. Data recorded includes the application name, Telnet LU name, client and host IP address and port, time of logon or logoff, and data count in and out. Combined with the SMF utility exit routine, SMF data can be used to track Telnet usage by a number of variables. If statements for both format types are

coded then both record types are written. That capability should be used sparingly due to the additional processing costs involved in generating both types records. Refer to *z/OS Communications Server: IP Configuration Reference* for SMF record layout.

TESTMODE

The TESTMODE parameter in TELNETPARMS allows a profile to be processed by Telnet but then dropped before it becomes an active profile. Using TESTMODE ensures that LU assignments, security levels, and other Telnet parameters are not compromised due to profile syntax errors.

DISPLAYS

Several displays are available which provide summary and detail information. Telnet provides Profile-related displays to present the following information. Refer to *z/OS Communications Server: IP System Administrator's Commands* for detailed examples.

General information:

- PROFILE - All parameter information, such as timers and security information.
- WLM - Summary of all workload manager names registered to represent Telnet.

Mapping information:

- Object - Various levels of Object information can be displayed.
  - A list of all Objects sorted by Object type can be generated by omitting the TYPE= and ID= parameters.
  - A detailed display of Objects showing how they are mapped to Client Identifiers can be generated by specifying the TYPE= parameter.
  - A detailed display of Objects showing how they are mapped to Client Identifiers and the details of the Object group can be generated by specifying the ID= parameter.
  - A detailed display showing where an Object is used and how those Objects are mapped to Client Identifiers can be generated by specifying the WHEREUSED and ID= parameters.
- Client Identifier - Various levels of Client Identifier information can be displayed.
  - A list of all Client Identifiers sorted by Client Identifier type can be generated by omitting the TYPE= and ID= parameters.
  - A detailed display of Client Identifiers showing how they are mapped to Objects can be generated by specifying the TYPE= parameter.
  - A detailed display of Client Identifiers showing how they are mapped to Objects and the details the Client Identifier group can be generated by specifying the ID= parameter.
  - A detailed display showing where a Client Identifier is used and how Objects are mapped to those Client Identifiers can be generated by specifying the WHEREUSED and ID= parameters.
- INACTLUS - Summary of all LUs that have been inactivated by the operator or by Telnet as a result of OPEN ACB problems.

Connection information:

- CONNECTION - Summary of all connections, filtered summary of all connections, or detailed data of a single connection.

The above displays give good snapshots of the state of Telnet and its connections. In addition, the WHEREUSED option is very useful as a lookup diagnostic tool. The

display will show where any name or IP address is used within the profile. This is particularly helpful when a connection request does not proceed as expected. For example, a client is sent a USSMSG10 screen instead of immediately getting in session with a logon manager. A WHEREUSED display of the client's IP address might show, as expected, the IP address within an IP group that is mapped to the logon manager as a default application. However, another WHEREUSED display of the client's host name might show the host name within an HN group that is mapped to a USS table. Knowing that the selection order of Client Identifiers will cause the USS table to be chosen, the administrator will realize there are conflicting mapping statements for the client and will be able to resolve the problem.

Module information: Telnet module storage can be displayed to verify the level of maintenance. For example, symptoms indicate a problem that has been fixed by an APAR. It is not known for sure if the fixed module is in the current version of Telnet. The module in question can be displayed using the TCP/IP display STOR command. In the example below no APAR is listed, indicating it has not been applied to the system.

```
d tcpip,,stor,mod=ezbtvxrc
  EZZ8456I TCPIP MODULE STORAGE
    EZBTVXRC LOADED AT 0A220328 IN LOAD MODULE EZBTTMST
    +0000 47F0F026 20C5E9C2 E3E5E7D9 C340F0F0 *.00..EZBTVXRC 00
    +0010 4BF0F0F5 40F0F17A F2F97AF2 F240C8E3 *.005 01.29.22 HT
    +0020 C3D7F5F0 C1000DC0 58300224 58403150 *CP50A........ ..
  EZZ8459I DISPLAY TCPIP STOR COMPLETED SUCCESSFULLY
```

CTRACE

If a problem is not resolved using the above tools, IBM service will likely need a CTRACE with option Telnet. CTRACE, with only the Telnet option, gives very complete information about the Telnet processes. To debug almost any Telnet problem no other CTRACE option is needed. Generally, the other options simply take up space creating a trace-wrap condition more quickly. If the problem is data related, use the FULLDATATRACE statement to trace all the data coming into and leaving Telnet rather than tracing only the first 64 bytes of data. FULLDATATRACE will cause a trace-wrap condition more quickly so should be set only if needed. It should be set in PARMSGROUP instead of TELNETPARMS if a subset of clients can be identified. For transform problems, the DBCSTRACE statement in TELNETPARMS or PARMSGROUP should be used to produce more trace entries in the SYSPRINT and TNDBCSER data sets.

WorkLoad Manager for Telnet (WLM)

Telnet can be part of a large sysplex environment where the server is replicated on many machines. One of the goals of such a system is to balance workload across the different machines. WLM is a method used to direct connection requests to various machines within the sysplex. See Chapter 10, "Domain Name System (DNS)" on page 417 for more details about WLM. The WLMCLUSTERNAME statement in TELNETPARMS is used by Telnet to specify WLM registration names. For example, assume Telnet resides on three machines.

- Machine 1 supports CICS and TSO.
- Machine 2 supports CICS, TSO, and IMS.
- Machine 3 supports CICS and IMS.

The best approach to implement multiple applications with WLM registration is to create a separate port for each application. For example, CICS can be assigned to port 223, TSO to port 323, and IMS to port 423. Each port number can be associated with a unique WLM clustername. Port 223 can be set up for CICS and have a WLM name of TNCICS. Port 323 can be set up for TSO and have a WLM

name of TNTSO. Port 423 can be set up for IMS and have a WLM name of TNIMS. By doing this, the WLM names are associated with ports rather than application names to a single port. This is important when multiple ports already exist on a Telnet system. If a port has a problem and must be quiesced, the WLM name associated with the port will be deregistered, blocking any new requests from coming to the disabled port.

```
TELNETPARMS       ; Machine 1
  PORT 223
  WLMCLUSTERNAME TNCICS ENDWLMCLUSTERNAME
ENDTELNETPARMS

TELNETPARMS       ; Machine 1
  PORT 323
  WLMCLUSTERNAME TNTSO ENDWLMCLUSTERNAME
ENDTELNETPARMS

BEGINVTAM
  PORT 223
  DEFAULTLUS  TCP0001..TCP0200..FFFFNNN  ENDDEFAULTLUS
  DEFAULTAPPL CICS
ENDVTAM

BEGINVTAM
  PORT 323
  DEFAULTLUS  TCP0001..TCP0200..FFFFNNN  ENDDEFAULTLUS
  DEFAULTAPPL TSO
ENDVTAM

;*************************************************

TELNETPARMS       ; Machine 2
  PORT 223
  WLMCLUSTERNAME TNCICS ENDWLMCLUSTERNAME
ENDTELNETPARMS

TELNETPARMS       ; Machine 2
  PORT 323
  WLMCLUSTERNAME TNTSO ENDWLMCLUSTERNAME
ENDTELNETPARMS

TELNETPARMS       ; Machine 2
  PORT 423
  WLMCLUSTERNAME TNIMS ENDWLMCLUSTERNAME
ENDTELNETPARMS

BEGINVTAM
  PORT 223
  DEFAULTLUS  TCP0201..TCP0400..FFFFNNN  ENDDEFAULTLUS
  DEFAULTAPPL CICS
ENDVTAM

BEGINVTAM
  PORT 323
  DEFAULTLUS  TCP0201..TCP0400..FFFFNNN  ENDDEFAULTLUS
  DEFAULTAPPL TSO
ENDVTAM

BEGINVTAM
  PORT 423
  DEFAULTLUS  TCP0201..TCP0400..FFFFNNN  ENDDEFAULTLUS
  DEFAULTAPPL IMS
ENDVTAM

;*************************************************

TELNETPARMS       ; Machine 3
```

```
  PORT 223
  WLMCLUSTERNAME TNCICS ENDWLMCLUSTERNAME
ENDTELNETPARMS

TELNETPARMS       ; Machine 3
  PORT 423
  WLMCLUSTERNAME TNIMS ENDWLMCLUSTERNAME
ENDTELNETPARMS

BEGINVTAM
  PORT 223
  DEFAULTLUS  TCP0401..TCP0600..FFFFNNN  ENDDEFAULTLUS
  DEFAULTAPPL CICS
ENDVTAM

BEGINVTAM
  PORT 423
  DEFAULTLUS  TCP0401..TCP0600..FFFFNNN  ENDDEFAULTLUS
  DEFAULTAPPL IMS
ENDVTAM
```

With this setup, an end user can connect to host TNCICS, TNTSO, or TNIMS and will automatically be directed to the correct machine for load balancing. If a port is disabled by a Quiesce or Stop on one machine, the port will be deregistered from WLM so that no new connection requests will be received. It is important to set up the WLM names as links to port numbers rather than links to applications. By creating a port per application, end users think they are specifying an application but in reality are specifying a port number.

If more than one port is used for connectivity to the same application, then two different WLM names should be used for the two ports. Perhaps TNCICS and TNCICS1 could be used. If the same WLM name is used for both ports, a problem will occur if one of the ports is disabled. Because a second port is still active for the single WLM name, the name is *not* deregistered for the machine. WLM will continue to direct connection requests to the disabled port, where they will be rejected.

Configuring the z/OS UNIX Telnet server (otelnetd)

The z/OS UNIX Telnet server provides access to z/OS UNIX shell applications on the host using the Telnet protocol.

Installation information

The HFS files used in the z/OS UNIX Telnet server and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file. otelnetd writes to syslog facility local1.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/otelnetd
This is a symbolic link to /usr/lpp/tcpip/sbin/otelnetd. /usr/lpp/tcpip/sbin/otelnetd is a sticky-bit file. The OTELNETD member of *hlq*.SEZALOAD contains the executable code for the Telnet server.

/etc/banner
This file contains a login message which will be printed to the client's screen unless the -h option is specified. This banner should be stored here.

/etc/utmpx
This file is updated by the call to fsumoclp. It contains a list of all the users who are logged in with their associated tty.

/dev/ptypXXXX and /dev/ttypXXXX
These special device files represent pseudoterminals (ptys); they are used by the z/OS UNIX Telnet server and other programs.

Note: For information on allocating more of these files for more connections, see *z/OS UNIX System Services Planning*.

/usr/share/lib/terminfo
The descriptions of supported terminals are stored here. For more information, see *z/OS UNIX System Services Planning*.

/usr/lib/nls/msg/C/tnmsgs.cat
The message catalog used by the z/OS UNIX Telnet server is stored here.

If the message catalog does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

/usr/man/C/cat1/otelnetd.1
This file contains the associated manual (man) pages for the z/OS UNIX Telnet server. It provides online help for the user.

Starting, stopping, and administration of z/OS UNIX Telnet

The z/OS UNIX Telnet server is started by inetd for each incoming Telnet connection. When the Telnet session is complete, the z/OS UNIX Telnet server will exit. Each active Telnet session will have a separate instance of the Telnet server which will communicate with the Telnet client.

The z/OS UNIX inetd daemon does not propagate environment variables other than PATH and TZ to its child processes, so the NLSPATH and LANG environment variables cannot be used to point to a different message catalog.

The following standards are supported:
- RFC 854 Telnet Protocol Specification
- RFC 855 Telnet Option Specification
- RFC 856 Telnet Binary Transmission
- RFC 857 Telnet Echo Option
- RFC 858 Telnet Suppress Go Ahead Option
- RFC 859 Telnet Status Option
- RFC 860 Telnet Timing Mark Option
- RFC 861 Telnet Extended Options - List Option
- RFC 885 Telnet End of Record Option
- RFC 1073 Telnet Window Size Option
- RFC 1079 Telnet Terminal Speed Option
- RFC 1091 Telnet Terminal type option
- RFC 1096 Telnet X Display Location Option
- RFC 1123 Requirements for Internet Hosts -- Application and Support
- RFC 1184 Telnet Linemode Option
- RFC 1372 Telnet Remote Flow Control Option
- RFC 1571 Telnet Environment Option Interoperability Issues

- RFC 1572 Telnet Environment Option
- RFC 2941 Telnet Authentication Option
- RFC 2942 Telnet Authentication: Kerberos Version 5
- RFC 2946 Telnet Data Encryption Option
- RFC 2952 Telnet Encryption: DES 64 bit Cipher Feedback
- RFC 2953 Telnet Encryption: DES 64 bit Output Feedback

When a z/OS UNIX Telnet session is started up, otelnetd sends Telnet options to the client side indicating a willingness to do the following:

- WILL ENCRYPT
- DO ENCRYPT
- DO TERMINAL TYPE
- DO TSPEED
- DO XDISPLOC
- DO NEW-ENVIRON
- DO ENVIRON
- WILL SUPPRESS GO AHEAD
- DO ECHO
- DO LINEMODE
- DO NAWS
- WILL STATUS
- DO LFLOW
- DO TIMING-MARK

The z/OS UNIX Telnet server can enable the following options locally.

- WILL BINARY

  This option indicates that the client is willing to send 8 bits of data, rather than the normal 7 bits of network virtual terminal data.
- WILL ECHO

  When the LINEMODE option is enabled, a WILL ECHO or WONT ECHO will be sent to the client to indicate the current state of terminal echoing. When terminal echo is not desired, a WILL ECHO is sent to indicate that Telnet will take care of echoing any data that needs to be echoed to the terminal, and then nothing is echoed. When terminal echo is desired, a WONT ECHO is sent to indicate that Telnet will not be doing any terminal echoing, so the client should do any terminal echoing that is needed.
- WILL LOGOUT

  When a DO LOGOUT is received, a WILL LOGOUT is sent in response and the Telnet session is shut down.
- WILL SGA

  This option indicates that it will not be sending IAC GA, the go ahead command.
- WILL STATUS

  Indicates a willingness to send the client, upon request, the current status of all Telnet options.
- WILL TIMING-MARK

  Whenever a DO TIMING-MARK is received, a WILL TIMING-MARK is the response. It is only used in kludge linemode support.
- WILL ENCRYPT

  Indicates a willingness to encrypt the data stream.

The z/OS UNIX Telnet server can enable the following options remotely.

- DO BINARY

Sent to indicate that Telnet is willing to receive an 8-bit data stream.

- DO ECHO

  If a WILL ECHO is received, a DONT ECHO will be sent in response.
- DO ENVIRON

  Indicates a desire to be able to request environment variable information. (See RFC 1408.)
- DO LFLOW

  Requests that the client handle flow control characters remotely.
- DO LINEMODE

  Supports requests that the client do line-by-line processing.
- DO NAWS

  Requests that the client inform the server when the window size changes.
- DO NEW-ENVIRON

  Indicates a desire to be able to request environment variable information. (See RFC 1572.)
- DO SGA

  Indicates that it does not need to receive IAC GA, the go ahead command.
- DO TERMINAL-TYPE

  Indicates a desire to be able to request the name of the type of terminal that is attached to the client side of the connection.
- DO TERMINAL-SPEED

  Indicates a desire to be able to request information about the speed of the serial line to which the client is attached.
- DO TIMING-MARK

  Only supported if the client responded with WONT LINEMODE. If the client responds with WILL TM, then it is assumed that the client will support kludge linemode. It is not used for any other purposes.
- DO XDISPLOC

  Indicates a desire to be able to request the name of the X Window System display that is associated with the Telnet client.
- DO AUTHENTICATION

  Indicates a willingness to receive authentication information for automatic login.
- DO ENCRYPT

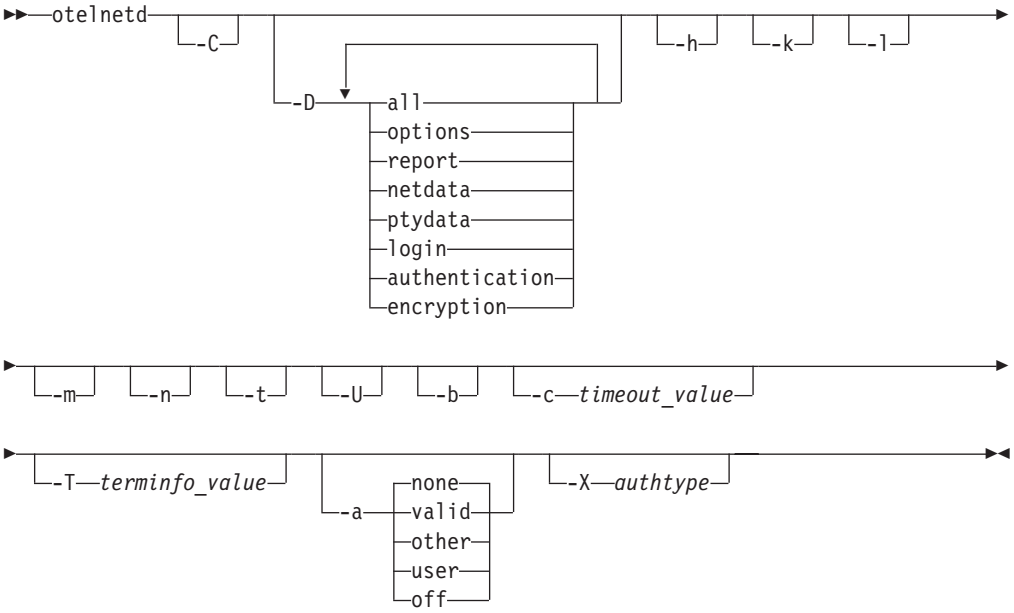
  Indicates a willingness to decrypt the data stream.

otelnetd

Note: The user ID associated with the daemon in /etc/inetd.conf requires superuser authority. Refer to *z/OS UNIX System Services Planning* for a description of the types of authority defined for daemons.

The following syntax is used in the /etc/inetd.conf file to define the arguments used to invoke otelnetd.

Syntax



Parameters

-C

Prints user messages in uppercase. There are several exceptions. Messages issued at startup are not affected by the -C option because the -C option is not processed during the startup. Also, data transmittal messages will not be uppercase. Data transmittal messages are generated from the -D netdata option or the -D ptydata option.

-D The following suboptions apply to -D:

options

Prints information about the negotiation of Telnet options. This information is used for debugging purposes. This suboption allows telnetd to generate debugging information to the connection, which allows the user to view telnetd activity.

report Prints the options information and additional information about processing. This information also includes print information designated for suboption=options. This can be used for debugging purposes. This suboption telnetd to generate debugging information to the connection, which enables the user to view telnetd activity.

netdata

Displays the data stream received by telnetd. This information is used

for debugging purposes. It allows telnetd to generate debugging information to the connection, which enables the user to view telnetd activity.

ptydata
Displays the data stream written to the pty. This information is used for debugging purposes. It allows telnetd to generate debugging information to the connection, which enables the user to view telnetd activity.

all Enables options, report, netdata, ptydata, login, authentication and encryption.

login Records login and logout activity to syslogd facility auth using message EZYTU36I.

authentication
Turns on authentication debugging code.

encryption
Turns on encryption debugging code.

-h Disables the display of the /etc/banner file at the user's terminal.

-k Disables kludge linemode. The server normally attempts to use kludge linemode when the -l option was specified, but the client does not support line mode. Use the -k option when there are remote clients that do not support kludge linemode, but pass the heuristic for kludge line mode support (for example, if they respond with WILL TIMING-MARK in response to a DO TIMING-MARK). This option does not disable kludge line mode when the client requests it. This is accomplished by the client sending DONT SUPPRESS-GO-AHEAD and DONT ECHO.

-l Specifies linemode, which tries to force clients to use linemode. If the LINEMODE option is not supported and the -k option was not specified, it will attempt to use kludge linemode.

Notes:

1. Many clients decline the server's request to operate in linemode.
2. Linemode is not appropriate for full-screen applications like the z/OS UNIX vi editor.

-m Enables the creation of a forked or spawned process to coexist in the same address space. This option can improve performance because the user's login shell runs in the same address space as otelnetd.

-n Disables TCP keep-alives. Normally, telnetd enables the TCP keep-alive mechanism to probe connections that have been idle for some time to determine if the client is still there. In this way, idle connections from machines that have crashed or can no longer be reached can be cleaned up. The cleanup of disabled connections is controlled by the presence of the KEEPALIVEOPTIONS statement in the TCPIP profile.

-t Specifies internal tracing. It also activates the REPORT option, as if the user also specified -D Report.

-U Causes telnetd to drop connections from any IP address that cannot be mapped back into a symbolic name by the gethostbyaddr or getnameinfo routines.

-b Forces the server to DO BINARY in the first pass during negotiations with the client.

-c *timeout_value*
Specifies the number of seconds to wait before terminating the Telnet session for inactive connections. The *timeout_value* is a value between 1 and 86400 seconds.

-T *terminfo_value*
Sets the TERMINFO environment variable to the specified values at startup. This option is needed when terminfo definitions are located in nonstandard directories.

-a This option may be used for specifying what mode should be used for authentication. There are several valid suboptions for authentication mode:

valid
Only allow connections when the remote user can provide valid authentication information to identify the remote user. Thus, for otelnetd, Kerberos authentication will be required. User verification will still occur through the login and password prompt. However, if the login user ID matches the name in the Kerberos principal, then no password will be requested. This is the most secure authentication mode.

other
Only allow connections that supply some authentication information. This option is currently not supported by any of the existing authentication mechanisms, and is thus the same as specifying -a valid.

user
Only allow connections when the remote user can provide valid authentication information to identify the remote user, and is allowed access to the specified account without providing a password. Thus, for otelnetd, Kerberos authentication is required. The NAME received during AUTHENTICATION option negotiation must match the name in the Kerberos principal and be a valid user ID on the host. No user verification will occur through the login or password prompt.

none
This is the default state. Authentication information is not required. User verification will still occur through the login and password prompt. However, if the login user ID matches the name in the Kerberos principal, then no password will be requested.

off
This disables the authentication code. All user verification happens through the login and password prompt. During option negotiation, otelnetd will not send DO AUTHENTICATION and, if necessary, will send DONT AUTHENTICATION.

Note: Authentication is not supported for IPv6 connections. If tcp6 is specified in inetd.conf, -a should not be used as a start option. If tcp6 and -a are both specified, the suboption will be overridden and forced to OFF.

-X *authtype*
This option disables the use of authtype authentication. Currently the only valid value for authtype is KERBEROS_V5. Thus, if otelnetd sends the AUTHENTICATION option SEND command, the authentication-type-pair-list will not contain any KERBEROS_V5 entries and will be empty.

SMF record handling

The SMF records generated are the typical set of records that MVS generates for start of job (login) and end of job (logoff). Additionally, interval records might be issued during the life of the user login. These records are SMF type 30 and type 72 and not the type 118 or type 119 in the current z/OS UNIX Telnet server. The process of issuing these records is external to the specific daemons.

BPX.DAEMON considerations

If the BPX.DAEMON facility class is defined, perform the following additional configuration steps:

1. Provide read access to BPX.DAEMON for the user ID specified in /etc/inetd.conf for otelnetd.
2. Define *hlq*.SEZALOAD to program control.
3. Define the C run-time library, hlq.SCEERUN to program control.

Refer to *z/OS UNIX System Services Planning* for more information about the BPX.DAEMON facility class, the security product commands used to perform the required configuration, and the diagnosis procedure for resolving related problems.

Kerberos

otelnetd supports Kerberos Version 5 for authentication on IPv4 connections. Authentication is not supported on IPv6 connections (that is, if tcp6 is specified for otelnetd in inetd.conf). On z/OS, Kerberos is implemented by Security Server. Please refer to*z/OS Security Server Network Authentication Service Administration* for more information.

The Kerberos principal used by otelnetd will generally be of the form ″host/<*hostname*>@*realm*″. That is, the first component of the Kerberos principal is ″host″; the second component is the fully qualified lowercase hostname of the server; and the realm is the Kerberos realm to which the server belongs.

otelnetd will not accept forwarded credentials from the client.

Successful AUTHENTICATION option negotiation is required for successful ENCRYPT option negotiation. The ENCRYPT option must be negotiated in both directions.

Chapter 9. Transferring files using FTP

The File Transfer Protocol (FTP) allows a user to copy files from one machine to another. The protocol allows for data transfer between the client (the end user) and the server in either direction. In addition to copying files, the client can issue FTP commands to the server to manipulate the underlying file system of the server (for example, to create or delete directories, delete files, rename existing files, and so on.) FTP is the most frequently used TCP/IP application for moving files between computers.

Copying files from one machine to another is one of the most frequently used operations. The data transfer between client and server can be in either direction. The client can send a file to the server machine. It can also request a file from this server.

To access remote files, the user must identify himself or herself to the server. At this point the server is responsible for authenticating the client before it allows the file transfer.

From an FTP user's point of view, the link is connection-oriented. FTP uses TCP as a transport protocol to provide reliable end-to-end connections. Both hosts must run TCP/IP to establish file transfer.

The z/OS model for the FTP server includes a daemon process and a server process. The daemon process starts when you start your cataloged procedure (for example, START FTPD) and it listens for connection requests on a specific port. The port is the well-known port 21 unless otherwise specified. For methods of choosing a different port number, see "Configuring ETC.SERVICES" on page 385 and "Configuring the FTPD cataloged procedure" on page 385. When the daemon accepts an incoming connection, it creates a new process (server's address space) for the FTP server, which handles the connection for the rest of the FTP login session. Each login session has its own server process.

The server process inherits the accepted connection from the daemon process. This connection is called the control connection. The server receives commands from the client and sends replies to the client using the control connection. The control connection port is the same as the daemon's listening port.

The client and server use a different connection for transferring data; this connection is called the data connection. By default, the data port is one less than the control connection port. For example, if the control connection port is 21, the data port is 20. An FTP client can override the default data port by directing the server to run in passive mode. In passive mode, the server uses an ephemeral port for the data port. Passive mode is requested by firewall friendly clients and by clients initiating three-way data transfers.

Note: If you use the environment variable _BPX_JOBNAME when you start FTPD, the server's address space is known as the jobname specified in the _BPX_JOBNAME variable. Having a common naming convention for your installation's FTP address spaces may be needed if your installation uses syslogd isolation or has other workload management requirements.

If you do not use _BPX_JOBNAME, the server's address space assumes the name of the user. For example, if a user logs into an FTP server with the user ID of TCP0001, the FTP server address space servicing the request is also known as TCP0001.

If the FTP daemon accepts a connection that is protected by the TLS security mechanism and you are not using _BPX_JOBNAME, the server's address space name is a name derived from the FTP server jobname. The name is of the form *jobnamex*, where *jobname* is the jobname, and *x* is a numeral from 1 to 9.

Configuring PROFILE.TCPIP for FTP

If you have configured the FTP server to have affinity to a specific stack, the FTP server can be started automatically when the TCP/IP address space is started by specifying the name of the FTP server cataloged procedure in the AUTOLOG statement. Also, if you have configured the FTP server to be a generic server in a single stack environment, you can use the AUTOLOG statement to automatically start the server. If, however, you configure the FTP server as a generic server in a multiple stack environment, you should not use the AUTOLOG statement to automatically start the server. Instead, use an operations automation software package to automatically start the server.

In the following example, if your procedure is called FTPD, the following statement allows TCP/IP to issue the MVS start command for procedure FTPD. The job name of FTPD1 will be used on the port statement shown below. If the daemon job name is less than eight characters, the FTP daemon forks() a process that has the job name of the original daemon appended with the number 1.

```
AUTOLOG
  FTPD JOBNAME FTPD1
ENDAUTOLOG
```

To reserve ports 21 and 20 for the FTP server, add the following:

```
 PORT
   21 TCP FTPD1             ; FTP server control port
   20 TCP OMVS NOAUTOLOG    ; FTP server data port
```

Specifying FTPD1 on the PORT and AUTOLOG statements directs TCP/IP to restart FTPD if it shound end.

To allow FTP to detect data connection errors when there has been no activity on the data connection for a certain amount of time, set the INTERVAL parameter on the TCPCONFIG statement to a relatively low value. The keepalive packets that the stack sends as specified on the INTERVAL parameter enable the stack to detect errors, such as a reset or terminated peer connection, instead of waiting indefinitely. Be careful when choosing an INTERVAL value on the TCPCONFIG statement because this value will affect all TCP connections at the host for which the interval has been activated, not just FTP connections.

The control connection can also benefit from keepalive packets. Many firewalls require periodic activity on any connection that is made and the control connection can appear idle during a long data transfer. Coding the INTERVAL parameter on the TCPCONFIG statement will, of course, cause keepalive packets to be sent on the control connection as well as the data connection. For FTP control connections only, you can override the keepalive interval you have configured in the stack with the FTPKEEPALIVE statement in FTP.DATA.

FTP requires a buffer size of 180K for data connections, therefore, you must not set TCPMAXRCVBUFRSIZE below 180K. The default value for the parameter is 256K.

For more information about the AUTOLOG, PORT, and TCPCONFIG statements, refer to *z/OS Communications Server: IP Configuration Reference*.

Configuring ETC.SERVICES

The ETC.SERVICES file contains the relationship between service names (servers) and port numbers in the z/OS UNIX environment. If necessary, update your ETC.SERVICES file to include the control port that the FTP server is to use. For the search order used to locate the ETC.SERVICES file, see "Configuration files for TCP/IP applications" on page 26. For example, add the following:

```
ftp 21/tcp
```

Notes:

1. In the ETC.SERVICES file, only one port (the one for the control connection) is listed.
2. The port specified for FTP in the ETC.SERVICES file can be overridden by the FTP start parameter, PORT nnnn, and should match the PORT statement in the PROFILE.TCPIP.
3. If the ETC.SERVICES file is changed such that a port other than 21 is specified, then that port will become the FTP port for that z/OS host.

Configuring /etc/syslog.conf

Note: For FTP syslog, you should consider the fact that FTP writes log messages to the system console if syslogd is not running. If you enable FTP server traces without syslogd active, large amounts of data might be written to the system console.

The daemon.*priority* entries in /etc/syslog.conf determine where FTP messages and trace entries are written. The FTP server issues info, warning, and error messages. All trace entries are written with debug priority. To direct trace entries (and all messages) to /tmp/daemon.trace, include the following in /etc/syslog.conf:

```
*.*.daemon.debug    /tmp/daemon.trace
```

Log messages can be isolated within syslogd. For FTP, an installation might want FTP log messages to be written to different files depending on the user ID, or separately for the FTP daemon. If FTP messages are to be isolated for user1, use the first statement below. If FTP messages are to be logged for all the FTP applications, use the second statement below.

```
user1.*.daemon.debug /tmp/daemon.trace
*.FTPD*.daemon.debug /tmp/daemon.trace
```

In the above statement, it is assumed that _BPX_JOBNAME is set to FTPD.

Configuring the FTPD cataloged procedure

Update the FTP cataloged procedure FTPD by copying the sample in *hlq*.SEZAINST(FTPD) to your system or recognized PROCLIB and modifying it to suit your local configuration. The EXEC PARM=, SYSFTPD DD, and SYSTCPD statements must be updated.

See “Configuring FTP.DATA” on page 389 to configure SYSFTPD DD and “Configuring TCPIP.DATA for FTP” on page 389 to configure SYSTCPD DD.

The system parameters required by the FTP server are passed by the PARM parameter on the EXEC statement of the FTPD cataloged procedure. Add your parameters to `PARMS='` in the PROC statement of the FTPD cataloged procedure, making certain that each parameter is separated by a blank and all parameters are in uppercase

For example: `//FTPD PROC MODULE='FTPD',PARMS='TRACE ANONYMOUS PORT 21'` tells FTP to start up with TRACE active, ANONYMOUS support enabled, and use control PORT 21.

Security considerations for the FTP server

Consider the following for security:

- User IDs

  To log into the FTP server, a user ID must have a z/OS UNIX UID or may use the default UNIX UID.
- MVS Network Access Controls
  - If PortAccess or NetAccess is used to SAF resource secure TCP ports or networks, see the NETACCESS statement in *z/OS Communications Server: IP Configuration Reference* for more information.
- The FTPD cataloged procedure must be:
  - Defined to the security program.
  - Added to the RACF started class facility or the started procedures table. The user ID associated with the FTP server started class must have a UID of 0.
  - See SEZAINST(EZARACF) for more information on SAF resource requirements needed for FTP.
- Terminal Access

  For IPv4 connection partners, the terminal ID passed from FTP to RACF is an 8-byte hexadecimal character string containing an IPv4 address. RACF interprets this as a terminal logon address and rejects it if it is not previously defined. For example, the IP address 163.97.227.17 is translated to X'A361E311'.

  Therefore, if the SETROPTS TERMINAL(NONE) setting is used in RACF, you must define profiles for the IP addresses in class TERMINAL to avoid problems when trying to FTP to MVS. You must translate all the IP addresses of any clients connecting to FTP servers to hexadecimal character strings and add them to the class TERMINAL.

  To allow access by all addresses starting with 163, define a profile for all addresses in the 163.97.227 subnet:

  ```
  RDEFINE TERMINAL A361E3* UACC(READ)
  ```

  If your RACF SETROPTS options are TERMINAL(READ), all terminals are allowed access to your system, and you do not have to add extra resource definitions to your RACF data base.

  For IPv6 connection partners, no terminal ID is passed from FTP to RACF. RACF does not validate the login address. All IPv6 connection partners are allowed by RACF.

  For more information, see *z/OS UNIX System Services Planning* and the *z/OS Security Server RACF Security Administrator’s Guide*.

- Clients may use your server to send random data to other servers.

  Any FTP client in PROXY mode with your FTP server could establish a data connection to any server listening to a port. This could be very disruptive to that server, because the client could then send a very large amount of unexpected data to it. Any malicious FTP client can attack or disrupt the server in a normal server-to-client connection by making the FTP server send a large amount of data to another application server that is listening to a specific port. Since the client itself is not sending the disruptive data, it is difficult to identify the client that is causing the problem. PORTCOMMAND, PORTCOMMANDPORT and PORTCOMMANDIPADDR statements are provided in FTP.DATA to prevent your server from being used in this way.

Table 18. PORTCOMMAND scenarios

When you want your server to...	Code the following statements in the server's FTP.DATA:	Note:
Reject all PORT or EPRT commands	PORTCOMMAND REJECT	Disabling PORT or EPRT commands prevents your server from being used to send random data to other servers. However, your server loses some ability to transfer data in PROXY mode. If a client sends a PORT or EPRT command to your server to set up a proxy transfer, your server will reject the command and the proxy transfer fails. If your client is not firewall friendly, and it does not implement the default port number and IP address for data transfer, that client cannot transfer files to and from your server.
Reject all PORT or EPRT commands that specify well-known ports (port numbers less than 1024)	PORTCOMMANDPORT NOLOWPORTS	When you specify this combination, your server cannot be used to send random data to servers listening on well-known ports. However, a rogue client could use your server to send random data to servers listening on other ports. The server still supports data transfer in PROXY mode.
Reject all PORT or EPRT commands that specify an IP address other than the client's own IP address.	PORTCOMMANDIPADDR NOREDIRECT	When you specify this combination, a client can request data transfer in PROXY mode only between your server and a server on its own IP address. Transfers between client and server are not affected.
Reject all PORT or EPRT commands that specify an IP address other than the client's own IP address or port numbers that are well known.	PORTCOMMANDPORT NOLOWPORTS PORTCOMMANDIPADDR NOREDIRECT	When you specify this combination, a client can request data transfer in PROXY mode only between your server and a server on its own IP address, and the port numbers cannot be well known. The client cannot use PROXY mode to send random data to a server on its own IP address listening to a well-known port.

- Using Generic Security Service Application Programming Interface(GSSAPI) to Authenticate Users

  GSS can be used to authenticate FTP clients to FTP servers. A client can attempt to authenticate to the server by sending a command specifying GSS as the authentication type. GSSAPI has Kerberos 5 as just one of many possible security services. For more information on setting up GSS support for the FTP server, refer to "Customizing the FTP server for the GSSAPI" on page 395.

- Connection Security

  The FTP server and client support TLS. This support enables secure file transfer by providing data privacy, message authentication, and message integrity services for data sent and received using the FTP control and data connections. Optionally, an FTP client certificate that is authenticated during the TLS handshake can be used for end user identification and authentication in addition to user ID and password validation. For more information on setting up TLS support for the FTP server, refer to “Customizing the FTP server for TLS” on page 394.

Defining environment variables for the FTP server (optional)

The FTP server optionally uses environment variables to identify the translate table data sets to be used for the control and data connections. These environment variables are used to override a default naming convention as described below. CCXLATE and XLATE statements will be ignored if EXTENSIONS UTF8 is specified in FTP.DATA.

_FTPXLATE_name used for translation

In your FTP.DATA file, you can use the CCXLATE or XLATE statements to specify a name that corresponds to a particular data set that is to be used for the initial translate tables for the control or data connections.

FTP will look for an environment variable defined as *_FTPXLATE_name=fully_qualified_dsn*, where *name* must be one to eight uppercase characters or numbers, and *fully_qualified_dsn* can be a fully qualified MVS data set name or HFS file name.

If the environment variable exists, FTP will use the data set name defined by the environment variable. If no such environment variable is defined, FTP will use the data set name *hlq.name*.TCPXLBIN.

Similarly, from any client you can issue SITE XLATE= to set the translate tables for the data connection for that particular FTP session. The FTP server will look for an environment variable called *_FTPXLATE_name*. If the environment variable does not exist, the server will look for a data set called *hlq.name*.TCPXLBIN.

Note: The CCXLATE and XLATE statements and SITE XLATE command are not case-sensitive, but the name of the optional environment variable is case-sensitive and must be in uppercase or FTP will not recognize it.

TZ and other UNIX environment variables

You can use the ENVAR runtime option in your FTPD start procedure to set environment variables for the FTP server. For information on using the ENVAR runtime option to set environment variables, see *z/OS C/C++ Programming Guide*. The following example shows how to specify environment variables in your FTPD started procedure:

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("TZ=EST")/&PARMS')
```

_BPX_JOBNAME

An installation that wants all FTP forked tasks to have similar job names needs to set the _BPX_JOBNAME environment variable. WorkLoad Manager (WLM), accounting, and isolation of syslogd messages as reasons an installation might not want to have each FTP logged-in user to have a job name of its user ID.

The following example sets all FTP forked() tasks to have the job name of FTPD:

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("_BPX_JOBNAME=FTPD"',
//           '"TZ=EST")/&PARMS')
```

_BPXK_SETIBMOPT_TRANSPORT for affinity to a specific stack

As discussed in “Generic server versus server with affinity for a specific transport provider” on page 55, if an installation wants to ensure that FTP has an affinity to a TCP/IP stack, the _BPXK_SETIBMOPT_TRANSPORT keyword should be used.

The example below sets the FTP server to have an affinity to TCPIEOE.

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=('POSIX(ON) ALL31(ON)',
//           'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPOE"',
//           '"TZ=EST")/&PARMS')
```

Configuring TCPIP.DATA for FTP

The following five statements are used by the FTP server:

DATASETPREFIX
Specifies HLQ for dynamic allocation

DOMAINORIGIN
Specifies the domain name to be appended to host name

HOSTNAME
Specifies the TCP host name

LOADDBCSTABLES
Specifies the DBCS tables used by the client and server

MESSAGECASE
Specifies the case that messages should be displayed in

See Chapter 1, “Configuration overview” on page 3 for information about TCPIP.DATA or refer to *z/OS Communications Server: IP Configuration Reference* for information about these statements.

Configuring FTP.DATA

The FTP.DATA data set is optional. The FTP daemon looks for this data set during initialization, following this sequence:

1. A data set specified by the //SYSFTPD DD statement
2. *ftpserve_job_name*.FTP.DATA
3. /etc/ftp.data
4. SYS1.TCPPARMS(FTPDATA)
5. *hlq*.FTP.DATA data set

It is not necessary to include all statements in the FTP.DATA data set. Only include the statements if the default value is not what you want, because the default will be used for any statement not included in the FTP.DATA data set.

Some FTP server parameters can be changed during an FTP session by the client issuing the SITE subcommand. See *z/OS Communications Server: IP User's Guide and Commands* for more information. The FTP client has an FTP.DATA data set which can also be used to change the defaults for the FTP client local site

parameters. See the *z/OS Communications Server: IP User's Guide and Commands* for more information about using the FTP.DATA data set for the FTP client local site parameters.

Optionally configuring user-level server options using FTPS.RC

The default values for the site parameters are coded in the server FTP.DATA. These SITE defaults apply to all login sessions to the server. You can customize settings for a specific user or group of users by creating an FTPS.RC configuration data set containing FTP commands specific to that login session. This file may contain a series of CWD and SITE commands. Refer to the *z/OS Communications Server: IP User's Guide and Commands* for information about these commands.

The FTP server uses the following search order to find the data set or HFS file:

1. tso_prefix.FTPS.RC
2. userid.FTPS.RC
3. $HOME/ftps.rc

Data set attributes

Data set attributes play a significant role in FTP performance. If your environment permits, tune both BLOCKSIZE and LRECL according to the following recommendations:

- Use half a track as the block size.
- For IBM 3380 DASD, use 23424 as the block size with an LRECL of 64 bytes.
- For IBM 3390 DASD or IBM9334, use 27968 as the block size with an LRECL of 64 bytes.
- Use FB as the data set allocation format.
- Use cached DASD controllers.
- If your environment permits, use a preallocated data set for FTP transfers into MVS.

The following configuration data statements apply to FTP server's allocation of data sets.

- AUTOMOUNT
- BLKSIZE
- BUFNO
- CONDDISP
- DATACLASS
- DCBDSN
- DIRECTORY
- LRECL
- MGMTCLASS
- MIGRATEVOL
- PRIMARY
- RECFM
- RETPD
- SECONDARY
- SPACETYPE
- STORCLASS
- UCOUNT

- UMASK
- UNITNAME
- VCOUNT
- VOLUME

Refer to *z/OS Communications Server: IP Configuration Reference* for more detailed information about these keywords.

Some of these allocation variables might provide duplicate information. FTP passes all variables that are specified to z/OS's dynamic allocation function and lets it determine which of the specifications take precedence. The only exceptions to this are the following:

- If the data set organization is physical sequential, then directory blocks are not sent.
- If neither primary nor secondary space quantities are specified, then the allocation units value is not sent.

For example, the model DCB (DCBDSN) might have a record format (RECFM) that differs from the record format specified by a data class and from the one explicitly specified by the client. The order of precedence for dynamic allocation variables are as follows:

1. Any FTP.DATA statements or SITE parameters explicitly specified or defaulted.
2. Any attributes picked up from the model DCB and not otherwise explicitly specified.
3. Any attributes picked up from the data class and not previously derived from 1 and 2 above.
4. Any system allocation defaults.

Specifying attributes for new MVS data sets

When allocating new data sets, there are two methods you can use to specify the data set attributes. You can individually use the data set attribute parameters with the SITE command or the statements in the FTP.DATA data set. Or, if your system programmer has used the Storage Management System to group together default attributes into named classes, you can specify those class names on the DATACLASS, STORCLASS, and MGMTCLASS statements.

Dynamic allocation

The FTP server allows a client program to dynamically allocate a new physical sequential data set or a partitioned data set (PDS) for the purpose of transferring data to be written to that data set. The following optional allocation variables can be used to override and turn off the defaults that affect the allocation of the data set.

Variable	**FTP.DATA statement**
allocation units	SPACETYPE
blocksize	BLKSIZE
data class	DATACLASS
directory blocks	DIRECTORY
logical record length	LRECL
management class	MGMTCLASS
model DCB values	DCBDSN
primary space	PRIMARY
secondary space	SECONDARY
unit count	UCOUNT
volume count	VCOUNT

record format	RECFM
retention period	RETPD
storage class	STORCLASS
unit	UNITNAME
volume serial number or list	VOLUME

Some of these allocation variables might provide duplicate information. For example, the model DCB might have a record format (RECFM) that differs from the record format specified by a data class and from the one explicitly specified by the client. FTP passes all variables that are specified to dynamic allocation and lets it determine which of the specifications take precedence. The following list describes the exceptions to that policy:

- If neither the primary nor secondary space quantity is specified, then the allocation units value is not sent.
- If the data set organization is physical sequential, then directory blocks specification is not sent.
- Otherwise, all variables are sent to dynamic allocation where the order of precedence is:
  1. Any FTP.DATA statements or SITE parameters explicitly specified or defaulted
  2. Any attributes picked up from the model DCB and not otherwise explicitly specified
  3. Any attributes picked up from the data class and not previously derived from 1 or 2
  4. Any allocation defaults

Storage Management Subsystem (SMS)

You can specify one or more of the following SMS classes to manage characteristics that are associated with or assigned to data sets.

- Data class is an SMS construct that an installation can define to control data set allocation attributes used by SMS for the creation of data sets. An installation can override all or part of an SMS DATA CLASS definition by using FTP.DATA statements. Note that there is an order of precedence for dynamic allocation. (See “Data set attributes” on page 390 for more information on the precedence.) The fields listed are available attributes that serve as a template for allocation. Each is *optional* and is overridden by any explicit specification of FTP allocation variables or by a model DCB (DCBDSN).

Variable	FTP.DATA statement
directory blocks	DIRECTORY
logical record length	LRECL
primary space	PRIMARY
record format	RECFM
retention period	RETPD
secondary space	SECONDARY

  Note: If either primary or secondary space is explicitly specified, then the primary and secondary values from data class are not used.

- Management class (MGMTCLASS) is an SMS construct that determines DFHSM action for data set retention, migration, backup, and release of allocated but unused space. Management class replaces and expands attributes that otherwise would be specified. That is, management class might override any other specification of retention period.

- Storage class (STORCLASS) is a list of storage performance and availability services requests for an SMS-managed data set that SMS attempts to honor when selecting a volume or volumes for the data set. It might conflict with an explicit specification of volume and unit. If storage class is used, then volume and unit should be unspecified.

Translation of data

Selecting an appropriate translate table for conversion of data from host to network format, and from network to host format, will ensure that data read from or written to the z/OS system are in correct format. The following statements apply to translation of data for the FTP server. Refer to *z/OS Communications Server: IP Configuration Reference* for more information on these statements. The statements are:

- ASATRANS
- CTRLCONN
- ENCODING
- EXTENSIONS UTF8
- MBDATACONN
- SBDATACONN
- SBSUB
- SBSUBCHAR
- UCSHOSTCS
- UCSSUB
- UCSTRUNC

Accounting

The following parameters apply to SMF data:

- SMF
- SMFAPPE
- SMFDEL
- SMFEXIT
- SMFJES
- SMFLOGN
- SMFREN
- SMFRETR
- SMFSQL
- SMFSTOR

Refer to *z/OS Communications Server: IP Configuration Reference* for more information on these statements.

Configure the FTP server for SMF (optional)

The FTP server can write SMF type 118 (X'76') or type 119 (X'77') records to record transactions made by the FTP server. SMF records can be written for the following commands:

- APPE (append)
- DELE (delete)
- RNTO (rename)
- RETR (retrieve)

- STOR (store)
- STOU (store unique)

Information about the previous commands can be recorded for:

- FTP server running in normal data transfer mode (FILETYPE=SEQ)
- FTP server running remote job submission (FILETYPE=JES)
- FTP server running Structured Query Language (SQL) queries (FILETYPE=SQL)
- Any combination of SEQ, JES, and SQL

For commands involving data transfer (APPEND, RETR, STOU or STOR) an SMF record will be written for both successfully and unsuccessfully completed data transfer commands which have begun data transfer. For data transfer commands which have completed unsuccessfully, the byte count of transmission field will contain the number of bytes transferred before the failure, and the recent server reply field will contain the 3-digit error reply code sent to the client. Refer to the *z/OS Communications Server: IP Configuration Reference* to find the particular offsets for the record type being used.

The FTP server can also write SMF records when a logon attempt fails.

The capability also exists for a user-written exit routine to get control before the SMF records are written. See “Configuring the optional FTP user exits” on page 396 for more information.

If you want the FTP server to write SMF type 118 (X'76') or type 119 (X'77') SMF records, you must include at least one of the SMF subtype statements (SMF, SMFAPPE, SMFDEL, SMFLOGN, SMFREN, SMFRETR, or SMFSTOR) in the FTP.DATA data set.

If SMF subtype statements are not coded in the FTP.DATA data set, no SMF records are written by the FTP server.

Customizing the FTP server for TLS

TLS is a level of security support that is added to both the FTP client and the FTP server. The client and server are configured with statements in the FTP.DATA file that specify the level of TLS required or the level of TLS allowed during an FTP connection. The FTP client also has start options that control the client’s behavior.

The following statements are available in FTP.DATA for the server:

- EXTENSIONS AUTH_TLS used to specify that the TLS authentication is supported.
- SECURE_FTP used to specify whether authentication is required.
- SECURE_LOGIN used to set the authorization level required for users.
- SECURE_DATACONN used to specify the minimum level of security allowed for the data connection.
- KEYRING used to specify the keyring file or resource name used for authentication
- CIPHERSUITE used to specify the name of a CipherSuite that is used for encryption and decryption.

The FTP client negotiates the level of security based on its parameters and the capabilities of the server. Likewise, the FTP server is configured for a certain level of support and must react to the client commands as appropriate.

There are two behaviors the client and server exhibit that provide the TLS support. They are:

- Client authentication

  This behavior occurs during the login process. It completes the FTP authorization process and determines the initial state of TLS protection for the control and data connections.
- Data connection

  This behavior determines whether the data connection is protected.

The control and the data connections are protected by TLS rules based on FTP.DATA file statements. The protection is established by the TLS negotiation (also known as the handshake) between the client and server. The FTP client program acts on the behalf of the end user of FTP to provide TLS enablement.

If the FTP server uses port 990 for its control connection, both the FTP client and server treat the control connection and the data connections as if they are protected by TLS. Port 990 is not required for a TLS connection to be established, only that when port 990 is used, TLS is assumed. Port 990 can be specified for the server as follows:

1. In the ETC.SERVICES data set, specify:

   ```
   ftp 990/tcp
   ```
2. In the procedure used to start the FTP server, specify the start parameter (option):

   ```
   PORT 990
   ```

Refer to the *z/OS Communications Server: IP Configuration Reference* for more detailed information.

Customizing the FTP server for the GSSAPI

The following statements are available in FTP.DATA for the server:

- EXTENSIONS AUTH_GSSAPI to specify that the Kerberos authentication type is supported using the GSSAPI.
- SECURE_FTP used to specify whether authentication is required.
- SECURE_LOGIN used to set the authorization level required for users.
- SECURE_CTRLCONN used to specify a minimum level of security allowed for the control connection.
- SECURE_DATACONN used to specify the minimum level of security allowed for the data connection.
- SECURE_PBSZ used to specify the maximum size of the encoded data blocks sent during file transfer.

GSSAPI authentication is supported for IPv4 connections only.

Refer to the *z/OS Communications Server: IP Configuration Reference* for more detailed information.

DB2® and JES

The following statements are used when FTP interfaces with DB2 and JES, respectively. For more information, refer to *z/OS Communications Server: IP Configuration Reference* and the optional steps in this chapter.

- DB2
- DB2PLAN
- SPREAD and SQLCOL
- JESLRECL
- JESPUTGETTO
- JESRECFM
- JESINTERFACELEVEL

Configuring the optional FTP user exits

The following describes exit routines you can code and install. For detailed information regarding these exit routines, refer to the *z/OS Communications Server: IP Configuration Reference*.

The FTPSMFEX user exit

Note the FTP server SMF user exit is called before an SMF type 118 record that contains information about an FTP server session is written to the SYS1.MANx data set. The user exit allows site specific modifications to the record and controls whether the record is written to the SYS1.MANx data set.

Note that the exit is called only for type 118 records. SMF type 119 FTP records must use the system-wide SMF user exits (IEFU83, IEFU84, and IEFU85) to obtain this same functionality. For information on these SMF user exits, refer to *z/OS MVS System Management Facilities (SMF)*.

The FTCHKIP user exit

The FTCHKIP user exit is called when a user attempts to log in to the FTP server or when a user issues the OPEN subcommand to establish a new connection. The following information is passed to the exit:

- Client IP address *
- Client port number *
- Server IP address *
- Server port number *
- The socket address structure of the client's control connection
- The socket address structure of the server's control connection
- Session instance identifier

Note: Fields above marked with an asterisk (*) are valid only for IPv4 addresses, including IPv4 addresses mapped into IPv6 format.

An installation can use this exit to determine if a particular IP address or port number is allowed to access the FTP site. If the connection is denied by the user exit, the following message is sent to the user:

```
421 User Exit rejects open for connection
```

The FTCHKPWD user exit

The FTCHKPWD user exit is called immediately after the user enters the password or e-mail address during login to the FTP server. The following information is passed to the exit:

- The user ID
- The user password or an asterisk (*) if an e-mail address is entered instead of a password
- A userdata buffer

  If an e-mail address is entered to log in, the userdata buffer contains the e-mail address.
- The number of bad passwords entering during this login attempt
- The socket address structure of the client's control connection
- The socket address structure of the server's control connection
- Session instance identifier

The exit can be used to restrict access to a site based on user ID, password, number of bad passwords, or anything in the socket address information for the client or server. If the login is denied by the user exit, the following reply is sent to the user:

```
530 PASS command failed
```

Note: If ACCCESSERRORMSGS TRUE is coded in FTP.DATA, an additional 530 reply with information about why the PASS command failed might precede the reply above.

The FTCHKCMD user exit

The FTCHKCMD user exit is called whenever the user enters an FTP command. The following information is passed to the user exit:

- The user ID
- The FTP command to be issued
- The command's arguments
- The directory type (MVS or HFS)
- The FILETYPE (SEQ, JES, or SQL)
- The current working directory
- A buffer to hold a modified argument string
- A buffer to hold a 500 reply extension to explain why the exit denied the request
- The socket address structure of the client's control connection
- The socket address structure of the server's control connection
- Session instance identifier
- A 256-byte scratchpad buffer

The user exit allows an installation to modify the arguments of an FTP command or to deny a user from issuing the command. For example, if a user issues a DIR * ftp command, the exit can either deny the command or modify it to DIR 'USER1.*'. If the user exit denies the request by this user to issue this command, one or both of the following replies will be sent to the user. The first reply is optional and is sent only if the user exit returns a string in the 500 reply extension buffer.

```
500-UX-buffercontents
500 User Exit denies Userid userid from using Command command
```

The FTCHKJES user exit

FTCHKJES is called if the server is in FILETYPE=JES mode and the client tries to submit a job. The following information is passed to the exit:

- The user ID
- A buffer containing the current JCL statement
- Size of statement in the buffer
- JESLrecl value
- Number of this buffer in current series
- Bytes transferred so far (including this buffer)
- Client identifier (see also session instance identifier)
- JESRecfm value
- FTCHKJES exit-specific workarea (4 bytes)
- The socket address structure of the client's control connection
- The socket address structure of the server's control connection
- Session instance identifier
- A 256-byte scratchpad buffer

The exit can allow or refuse the job to be submitted to the JES internal reader based on any information passed to the exit. For example, the exit can look for a USER= parameter on the JOB statement and check it against the client's user ID. If the remote job submission is denied, the exit sends the user the following reply:

```
550 User Exit refuses this job to be submitted by userid
```

The FTPOSTPR user exit

FTPOSTPR is called after execution of the FTP commands RETR, STOR, STOU, APPE, DELE, and RNTO. The following information is passed to the exit:

- The user ID
- Client IP address *
- Client port number *
- The directory type (MVS or HFS)
- The current working directory
- The FILETYPE (SEQ, JES, or SQL)
- Most recent reply code number
- Most recent reply text string
- Current FTP command
- Current CONDDISP setting
- Close reason code
- Name of data set or HFS file retrieved or stored
- Bytes transferred
- The socket address structure of the client's control connection
- The socket address structure of the server's control connection
- Session instance identifier
- A 256-byte scratchpad buffer

Note: Fields above marked with an asterisk (*) are valid only for IPv4 addresses, including IPv4 addresses mapped into IPv6 format.

The exit allows for post processing at the termination of data transfer functions within the server.

Customizing the FTP-to-JES interface for JESINTERFACELevel 2 (optional)

If FTP.DATA does not change the JESINTERFACELEVEL to 2, the FTP server uses the JES interface provided in releases prior to CS for OS/390 V2R10. At this level, the FTP user is allowed to submit jobs to JES, retrieve held output matching their logged-in user ID plus one character, and delete held jobs matching their logged-in user ID plus one character.

If JESINTERFACELevel is set to 2, then FTP users have the ability to retrieve and delete any job in the system permitted by the System Authorization Facility (SAF) resource class JESSPOOL. For that reason, JESINTERFACELevel=2 should only be specified if the proper JES and SDSF security measures are in place to protect access to JES output. The SAF controls used for JESINTERFACELevel=2 are essentially a subset of those used by SDSF. Therefore, if an installation has customized SAF facilities for SDSF, then they are configured for FTP JES level 2.

Before customizing the FTP-to-JES interface, complete JES customization. For example, JESJOBS is an SAF class that controls which users can submit jobs to JES. JESSPOOL is the SAF class that controls which users can access output jobs. Customize these SAF classes before beginning customization of the FTP-to-JES interface.

JESSPOOL defines resource names as <nodeid>.<userid>.<jobname>.<Dsid>.<dsname>. An FTP user can delete an output job if they have ALTER access to the resource that matches their nodeid, userid, and job name. If the FTP user has UPDATE access to the resource, they can list, retrieve, or GET the job output. (JESINTERFACELevel 2 uses the SAPI interface to JES, so UPDATE authority is required to list job status or retrieve job output.) For more information on JES security, refer to *z/OS JES2 Initialization and Tuning Guide*. For more information on the SAPI interface, refer to *z/OS MVS Using the Subsystem Interface*.

There are three filters used by the FTP server to control the display of jobs:

- JESSTATUS
- JESOWNER
- JESJOBNAME

SDSF resources are employed for this.

JESSTATUS can be changed by an FTP user with the SITE command to filter jobs in INPUT, ACTIVE, or OUTPUT state. The SDSF resources checked for these states are ISFCMD.DSP.INPUT.jesx, ISFCMD.DSP.ACTIVE.jesx, and ISFCMD.DSP.OUTPUT.jesx, respectively. At login time (USER command), the default value is set to ALL if READ access is allowed to all three classes. Otherwise it attempts to set it to OUTPUT, ACTIVE, and then INPUT if the appropriate READ access is allowed. If no READ access is allowed to any of the classes, JESSTATUS is set to OUTPUT but JESOWNER and JESJOBNAME cannot be changed from the default. In this way, SAF controls can be put in place to limit FTP users to whatever status of jobs an installation requires.

At login time, JESOWNER will have the value of the logged-in user ID. Authority to change JESOWNER is obtained through READ access to RACF profile ISFCMD.FILTER.OWNER. An FTP user who has READ access to ISFCMD.FILTER.OWNER will be allowed to change the JESOWNER parameter with the SITE command.

At login time, JESJOBNAME will have the value of the logged-in user ID plus an asterisk (*). Authority to change JESJOBNAME is obtained through READ access to RACF profile ISFCMD.FILTER.PREFIX. An FTP user who has READ access to ISFCMD.FILTER.PREFIX will be allowed to change the JESJOBNAME parameter with the SITE command.

For example, to allow all users except USER1 to be allowed to change JESOWNER enter the following:

```
SETROPTS CLASSACT(SDSF) REFRESH
RDEFINE SDSF (ISFCMD.FILTER.OWNER) UACC(READ)
PERMIT ISFCMD.FILTER.OWNER ACCESS(NONE) CLASS(SDSF) ID(USER1)
SETROPTS CLASSACT(SDSF) REFRESH
```

For more information on SDSF security, refer to *z/OS SDSF Operation and Customization*.

Configuring the FTP server for anonymous logins (optional)

You can configure the FTP server to accept anonymous logins. A login is anonymous when the remote user specifies USER ANONYMOUS instead of an FTP user ID. To enable anonymous logins, add the ANONYMOUS statement to the server FTP.DATA data set.

You can specify three levels of anonymous support via the ANONYMOUSLEVEL keyword. ANONYMOUSLEVEL 1 is the default, and is equivalent to anonymous login support provided by releases prior to OS/390 V2R10. That is, the ANONYMOUS statement is supported. If no operands are specified on the ANONYMOUS statement, the anonymous user needs no password and has unrestricted access to the MVS and HFS file systems.

You can specify ANONYMOUSLEVEL 2, but this is not recommended. ANONYMOUSLEVEL 2 is provided for migration purposes only. Consider ANONYMOUSLEVEL 3 if ANONYMOUSLEVEL 1 does not meet your anonymous login security requirements.

If you specify ANONYMOUSLEVEL 3, the anonymous user cannot issue the USER command to leave anonymous mode, nor can another user issue USER anonymous to enter anonymous login mode. If you specify ANONYMOUSLEVEL 3 and STARTDIRECTORY HFS in FTP.DATA, the anonymous user's HFS access is restricted to the anonymous user's home directory and home directory subtrees.

The ANONYMOUSLEVEL 3 server recognizes additional keywords that restrict the anonymous user's access to FTP resources. These keywords are ignored when ANONYMOUSLEVEL is less than three:

- ANONYMOUSFILEACCESS allows the system programmer to preclude access to either the HFS or MVS file systems.
- ANONYMOUSFILETYPEJES, ANONYMOUSFILETYPESQL, and ANONYMOUSFILETYPESEQ control whether the anonymous user can set filetype JES, SQL, or SEQ, respectively.

- ANONYMOUSHFSFILEMODE defines the mode bits used for files written to the HFS.
- ANONYMOUSHFSDIRMODE defines the mode bits used for directories created in the HFS.

Finally, when ANONYMOUSLEVEL is set to three, the user's e-mail address is requested in lieu of a password when:

- ANONYMOUS is specified without any parameters.
- ANONYMOUS is specified with user ID/password.
- ANONYMOUS is specified with user ID/SURROGATE.

Control the degree of verification of the e-mail address an anonymous user enters as password by using the EMAILADDRCHECK keyword in FTP.DATA. Refer to *z/OS Communications Server: IP Configuration Reference* for details about the EMAILADDRCHECK keyword. The e-mail address entered is logged to the syslog daemon and is also passed to a user exit routine, FTCHKPWD, for user processing.

The FTP server can be defined to process users without passwords by using the ANONYMOUS SURROGATE support. In order to support this, ANONYMOUSLEVEL must be set to 3 in FTP.DATA on the server and BPX.SRV surrogate must be defined in RACF.

z/OS UNIX uses profiles defined to the RACF SURROGAT class to authorize the server to act as a surrogate of a client. Profiles defined to the SURROGAT class are of the form:

```
BPX.SRV.<userid>
```

in which <userid> is the MVS user ID of the user that the server will support without a password.

The steps below are for a sample userid of the FTP daemon (the userid associated with the FTP started task procedure) called FTPD with the ability to support user ID GUEST without a password. As you add more servers, you will need to follow similar procedures.

1. Activate the SURROGAT class support in RACF:

   ```
   SETROPTS CLASSACT(SURROGAT)
   ```

   This has to be done only once on the system. The SURROGAT class may already have been set up on your system. If a daemon or server you are running will be using the SURROGAT support heavily, consider using the RACLIST command to keep the SURROGAT profiles in storage. The following example shows how to cache the SURROGAT profiles in storage:

   ```
   SETROPTS RACLIST(SURROGAT)
   ```

2. If the SURROGAT profile is in the RACLIST, any changes to the SURROGAT profiles must be followed by a REFRESH command. To create the SURROGAT class profile for user ID GUEST, issue:

   ```
   RDEFINE SURROGAT BPX.SRV.GUEST UACC(NONE)
   SETROPTS RACLIST(SURROGAT) REFRESH
   ```

   A similar SURROGAT profile is required for each user ID that a server must support without a password.

3. To permit the userid of the FTP daemon (the userid associated with the FTP started task procedure), FTPD, to create a security environment for user ID GUEST, issue the PERMIT command:

```
PERMIT BPX.SRV.GUEST CLASS(SURROGAT) ID(FTPD) ACCESS(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

If you choose ANONYMOUSLEVEL greater than one and you choose STARTDIRECTORY HFS, you must create an anonymous directory structure in the HFS.

Creating an anonymous directory structure in the HFS

The sample shell script, ftpandir.scp, will create an anonymous directory structure for you, containing required and optional structures. Or, a superuser can create the anonymous directory structure. In this section, the steps a superuser would follow to create an anonymous HFS directory structure are outlined.

For the following steps, assume that the RACF user ID that is used when an anonymous user logs in is called GUEST, that the HOME directory in that user's OMVS segment in RACF is /u/guest, and that FTP.DATA contains a statement similar to this: ANONYMOUS GUEST

1. Create a bin subdirectory in the anonymous root containing the executables *ls* and *sh*. This is a required directory. *ls* can be copied from the standard directory. *sh* is part of the standard MVS search order, so you need only create an empty file with the sticky bit.

   The following example shows how to create ls and sh in the user GUEST's home directory:

   ```
   ===> cd /u/guest
   ===> mkdir bin
   ===> chmod 711 bin
   ===> cd bin

   ===> cp /bin/ls ls
   ===> chmod 711 ls
   ===> touch sh
   ===> chmod 711 sh
   ===> chmod +t sh
   ```

   An ls -al command should give the following results. Owner and group attributes may be different in your system.

   ```
   # ls -al
   total 280
   drwx--x--x   2 USER22   0  8192 Sep 21 17:39 .
   drwx--x--x   7 USER22   0  8192 Nov  1 14:44 ..
   -rwx--x--x   1 USER22   0  126976 Sep 21 17:39 ls
   -rwx--x--t   1 USER22   0  0 Sep 21 17:39 sh
   ```

2. Create a usr/sbin sudirectory of the anonymous root containing the executable file ftpdns. This is a required subdirectory. The file ftpdns can be empty with the sticky bin on.

   The following example is for anonymous user GUEST:

   ```
   ===> cd /u/guest
   ===> mkdir usr
   ===> chmod 711 usr

   ===> cd usr
   ===> mkdir sbin
   ===> chmod 711 sbin
   ===> cd sbin
   ===> touch ftpdns
   ===> chmod 711 ftpdns
   ===> chmod +t ftpdns
   ```

If you do not configure the subdirectories, bin and usr/sbin, and their contents correctly, the FTP server will not be able to accept anonymous logins and message EZYFT731 will be displayed.

3. Create a dev subdirectory within the anonymous root. This is a required subdirectory. A null file is created in this directory and used during the open of syslog.

   The following example is for anonymous user GUEST:

   ```
   ===> cd /u/guest
   ===> mkdir dev
   ===> chmod 711 usr
   ```

   If you do not have the dev subdirectory, syslog might not open correctly. Messages such as EZA2830I will not be logged out correctly.

4. Set up the public directory structure. This is a required directory.

   This is the directory structure into which you place files that can be downloaded by the anonymous FTP user. It does not have to be named pub; it can be any name you choose. A general convention for anonymous FTP sites is to call it pub:

   ```
    ===> cd /u/guest
   ===> mkdir pub
   ===> cd pub
   ```

   If you want to structure the files you allow to be accessed, you can create multiple subdirectories underneath this directory.

   For simplicity, assume a single level directory, the pub directory. Into this directory you copy the files you want to allow the anonymous user to download:

   ```
   ===> cp /x/y/z/prodinfo1.txt prodinfo1.txt
   ===> cp /x/y/z/prodinfo2.txt prodinfo2.txt
   ===> cd ..
   ```

   Make sure that the permission bits are set correctly by using the following shell command when executed in the /u/guest directory. This will set the permission bits of all files in the pub directory and its subdirectories to 755:

   ```
   ===> chmod -R 755 pub
   ```

   If your system does not require an incoming or extract directory, the system is configured for anonymous FTP. An ls -al command of the pub directory should give the following results:

   ```
   drwxr-xr-x   3 IBMUSER  SYS1  8192 May 13 21:15 .
   drwxr-xr-x   6 IBMUSER  SYS1  8192 May 20 14:51 ..
   -rwxr-xr-x   1 IBMUSER  SYS1  12 May 11 12:41 prodinfo1.txt
   -rwxr-xr-x   1 IBMUSER  SYS1  12 May 11 12:41 prodinfo2.txt
   ```

5. Set up an incoming directory (optional).

   If you want anonymous users to be able to upload files to your FTP server, you need some additional setup. The objective is to allow an anonymous user to upload a file, but not to allow another anonymous user to download or even be aware of the existence of the file until after an administrative user has verified that the content of the file is acceptable. You do not want your FTP server site to become a store-and-forward site for files of questionable ethical content.

   Positioned at the /u/guest directory, a superuser issues the following shell command:

   ```
   ===> cd /u/guest
   ===> mkdir incoming
   ===> chmod 733 incoming
   ```

It does not have to be named incoming; it can be any name you choose. A general convention for anonymous FTP sites is to call it incoming.

The 733 permission bits means that a non-superuser cannot list the content of the incoming directory, but can write a file to it. Because the FTP server enforces a UMASK of 777 when an anonymous user logs in, these files will be written with permission bits 000, which means that they cannot be accessed by the anonymous user or by any other user except a superuser.

An FTP client user can normally change the UMASK via a SITE UMASK command or the user can change the permission bits of files they own through a SITE CHMOD command.

If you define ANONYMOUSLEVEL 3, you can use the ANONYMOUSHFSDIRMODE keyword to set the permission bits of any directory created by an anonymous user, and the ANONYMOUSHFSFILEMODE to set the permission bits of any file created by an anonymous user.

If you do allow anonymous users to store files on your FTP server, you should ensure that the directory into which these files are stored is in an HFS that can fill up without impacting other work on your z/OS system. The best way to do that is to allocate the /u/guest/incoming directory in its own HFS data set. If an anonymous user uploads large amounts of data to the incoming directory, only this separate HFS will be filled up. Filling this separate HFS will prevent other anonymous users from storing new files on the server, but will not affect other functions on your system. At a minimum, you should make sure that the incoming directory is not in the same HFS as your /tmp directory.

6. Set up the extract directory (optional).

   If you need to make files available to certain anonymous users, but not to everyone, you can create a directory that cannot be listed, but files in it can be downloaded if the anonymous user knows the name of the file.

   Positioned at the /u/guest directory, a superuser issues the following shell commands:

   ```
   ===> cd /u/guest
   ===> mkdir extract
   ===> chmod 711 extract
   ```

   It does not have to be named extract; it can be any name you choose. A general convention for anonymous FTP sites is to call it extract.

   A superuser can then copy files into this directory, ensure they have permissions of 755, inform the intended anonymous user of the file name, and that user can then log on as anonymous and retrieve the file.

   An ls -al command at the /u/guest location should give the following result, if you created all four subdirectories:

   ```
   drwxr-xr-x   6 IBMUSER  SYS1 8192 May 20 14:51 .
   dr-xr-xr-x   6 IBMUSER  SYS1  0 Jun 10 15:43 ..
   drwx--x--x   2 IBMUSER  SYS1  8192 May 11 12:44 bin
   drwx--x--x   3 IBMUSER  SYS1  8192 May 11 13:39 extract
   drwx-wx-wx   3 IBMUSER  SYS1  8192 May 25 09:35 incoming
   drwxr-xr-x   3 IBMUSER  SYS1  8192 May 13 21:15 pub
   ```

Configure the Welcome Banner Page, Login, and Directory Message (optional)

Starting in V2R10, the FTP server now provides support to allow FTP administrators to provide useful information about the site to FTP users. The following FTP.DATA statements are available:

- BANNER
- LOGINMSG
- ANONYMOUSLOGINMSG
- MVSINFO
- ANONYMOUSMVSINFO
- HFSINFO
- ANONYMOUSHFSINFO

You can use the LOGINMSG statement in FTP.DATA to point to a set of messages displayed when a known user logs in to FTP. Similarly, ANONYMOUSLOGINMSG can point to a set of messages displayed when an anonymous user logs in to FTP.

You can use the MVSINFO statement to point to a set of messages displayed when a known user changes the working directory to a particular MVS data set path. Likewise, use the ANONYMOUSMVSINFO statement to point to a set of messages displayed when an anonymous user changes working directory to a particular MVS data set path.

You can use the HFSINFO statement to point to a set of messages displayed when a client changes the working directory to a particular HFS directory. Likewise, use the ANONYMOUSHFSINFO statement to point to a set of messages displayed when an anonymous user changes working directory to a particular HFS directory.

Using magic cookies to represent information

The content of all the informational messages may include a predefined set of magic cookies, which are substituted by the FTP server before the data is sent to the FTP client. The following magic cookies are supported:

- %T — Local time
- %C — Current working directory
- %E — The FTP server administrators e-mail address
- %R — Remote host name
- %L — Local host name
- %U — Username (logged in user)

If %R is used, a long delay in login processing might occur as the FTP server will issue a DNS query to resolve the remote host IP address. In order to use %E, the ADMINEMAILADDR keyword must be specified in the server FTP.DATA configuration file.

Configuring to send detailed login failure replies to an FTP client (optional)

The FTP server returns minimal information to the client when the PASS command fails. However, you can configure the FTP server to send additional information by coding ACCESSERRORMSGS TRUE in FTP.DATA. This directs the server to reply to the client with detailed login failure data. The reply might report server errors, such as failing function calls with diagnostic return codes. It might report user errors, such as an expired or incorrect password, an unknown user ID, or a revoked user ID. You should not code ACCESSERRORMSGS TRUE in FTP.DATA if you do not want to share this type of information with users logging in to FTP.

You can capture the same information in syslog by coding FTPLOGGING TRUE and ANONYMOUSFTPLOGGING TRUE in FTP.DATA. You can also turn on the DEBUG option called ACC to log the error messages in the syslog. For more information on coding FTP.DATA statements, see *z/OS Communications Server: IP Configuration Reference*.

Install the SQL query function (optional) and access the DB2 modules

To use FTP to do SQL queries, bind the DBRM called EZAFTPMQ to the plan used by FTP, and grant execution privileges for that plan to PUBLIC. (The name of the plan can be specified by the DB2PLAN keyword in FTP.DATA or defaulted to EZAFTPMQ.) This FTP facility only performs SELECT operations on the DB2 tables. It does not perform UPDATE, INSERT, or DELETE.

Note: If secondary authorization for SQL queries is required, the DSN3SATH sample exit shipped by DB2 must be modified. The exit will return the primary AUTHID for requests originating from the FTP server.

The following sample job is provided in the FTOEBIND member of the SEZAINST data set. It can be used to enable the FTP server and client to do SQL queries.

```
//FTPSETUP JOB FTPSETUP,
//            CLASS=A,
//            NOTIFY=&SYSUID
//*************************************************************
//*
//*   File name:                 tcpip.SEZAINST(FTOEBIND)
//*   SMP/E distribution name:   EZAFTPAB
//*
//*    Licensed Materials - Property of IBM
//*    This product contains "Restricted Materials of IBM"
//*    5647-A01 (C) Copyright IBM Corp.  1997, 2002
//*    All rights reserved.
//*    US Government Users Restricted Rights -
//*    Use, duplication or disclosure restricted by GSA ADP Schedule
//*    Contract with IBM Corp.
//*    See IBM Copyright Instructions.
//*
//*  This JCL binds the EZAFTPMQ DBRM to the specified
//*  DB2 subsystem and allows execution of the
//*  EZAFTPMQ plan by PUBLIC.
//*
//*  The FTP server and client use this plan. (See
//*  Usage note #7)
//*
//*
//*  Usage notes:
//*
//*    1.  You must execute this job from a user ID that has
```

```
//*        the authority to bind the EZAFTPMQ plan.
//*
//*    2.  Change the STEPLIB DD statement in the FTPBIND and
//*        FTPGRANT steps to reflect the DB2 DSNLOAD data set.
//*
//*    3.  Change the DB2 sybsystem name in the FTPBIND and
//*        FTPGRANT steps from SYSTEM(xxx) to the
//*        installation defined DB2 subsystem name.
//*
//*    4.  Change the library parameter in the FTPBIND step from
//*        TCPIP.SEZADBRM to the installation defined TCPIP
//*        SEZADBRM library.
//*
//*    5.  Change the plan name in the FTPGRANT step from
//*        DSNTIAYY to reflect the plan associated with the
//*        program DSNTIAD.
//*
//*    6.  Change the library parameter in the FTPGRANT step
//*        from xxxxxx.RUNLIB.LOAD to reflect the library
//*        where the DSNTIAD program resides.
//*
//*    7.  You can bind the DBRM to a plan name other than EZAFTPMQ
//*        by changing the plan specified in the FTPBIND and
//*        FTPGRANT steps.  If you do this, you must use the
//*        DB2PLAN keyword in FTP.DATA to change the plan name
//*        used by the FTP server and/or client to the plan name
//*        specified here.
//*
//*************************************************************
//FTPBIND  EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN=xxxxxx.DSNLOAD,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSTSIN  DD   *
 DSN  SYSTEM(xxx)
 BIND ACQUIRE(USE) -
      ACTION(REPLACE) -
      CACHESIZE(1024) -
      CURRENTDATA(NO) -
      EXPLAIN(NO) -
      ISOLATION(CS) -
      LIBRARY('TCPIP.SEZADBRM') -
      MEMBER(EZAFTPMQ) -
      NODEFER(PREPARE) -
      PLAN(EZAFTPMQ) -
      RELEASE(COMMIT) -
      VALIDATE(RUN) -
      RETAIN
 END
//*
//FTPGRANT EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD   DSN=xxxxxx.DSNLOAD,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSTSIN  DD   *
 DSN  SYSTEM(xxx)
 RUN  PROGRAM(DSNTIAD) -
      PLAN(DSNTIAYY) -
      LIBRARY('xxxxxx.RUNLIB.LOAD')
 END
//SYSIN    DD   *
 GRANT EXECUTE ON PLAN EZAFTPMQ TO PUBLIC;
//*
```

Accessing DB2 modules

The FTP server or client loads 3 DB2 modules into storage to perform an SQL query. These modules are:

- DSNALI
- DSNHLI2
- DSNTIAR

The modules are usually found in the DB2 load library with the suffix DSNLOAD. The DB2 administrator or system programmer should add the DSNLOAD library to the LINKLIST to ensure FTP has access to this library.

Another way to ensure access is to add the DSNLOAD library to the FTP STEPLIB. For the FTP server this means the JCL used to start the FTP server has a STEPLIB DD statement referring to the DSNLOAD library or, if the FTP daemon is started from the z/OS shell, the STEPLIB environment variable is set. For the FTP client, this means a TSO CLIST must allocate the DSNLOAD library as the STEPLIB.

If the FTP client is to be run from a batch job to perform SQL queries, the DSNLOAD library must be added to the STEPLIB DD statement for the batch job.

Usage notes:

To allow FTP access to multiple levels of DB2, link to the libraries that contain the lowest level of DB2 to be accessed.

FTP.DATA updates for SQL query function

To obtain FTP.DATA updates for the SQL query function, follow these steps:

1. Set the FTP.DATA DB2 statement to specify the name of the DB2 subsystem.
2. Set DB2PLAN to specify the DB2 plan to be used by the FTP server.
3. Set the SPREAD statement to specify whether SQL output is in spreadsheet format.
4. Set SQLCOL to specify the column headings of the output data.

Trivial File Transfer Protocol (TFTP)

TFTP is a TCP/IP protocol used to transfer files. TFTP can read or write files from or to a remote server. On the z/OS system, TFTP is a server you can configure with the command line option during TFTP invocation.

Considerations for z/OS

TFTP is installed in the /usr/lpp/tcpip/sbin/ directory.

CAUTION:
The TFTP server uses well-known port 69. The TFTP server has no user authentication. Any client that can connect to port 69 on the server has access to TFTP. If the TFTP server is started without a directory, it allows access to the entire HFS. To restrict access to the HFS, start the TFTP server with a list of directories.

To start the TFTP server from the command line, type the tftpd command.

```
tftpd [-l] [-p port] [-t timeout] [-r maxretries] [-c concurrency_limit]
   [-s maxsegsize] [-f file] [-a archive directory [-a ...]]
   [directory ...]
```

The following are parameters used for the tftpd command:

-l Logs all incoming read and write requests and associated information to the system log. Logged information includes the IP address of the requestor, the file requested, and whether the request was successful.

-p port
Uses the specified port. The TFTP server usually receives requests on well-known port 69. You can specify the port in which requests are to be received.

-t timeout
Sets the packet timeout. The TFTP server usually waits 5 seconds before assuming a transmitted packet has been lost. You can specify a different timeout period in seconds.

-r maxretries
Sets the retry limit. The TFTP server usually limits the number of retransmissions it performs due to lost packet to 5. You can specify a different retry limit.

-c concurrency_limit
Sets the concurrency limit. The TFTP server spawns both threads and processes to handle incoming requests. You can specify the limit for the number of threads that may be concurrently processing requests under a single process. When the limit is exceeded, a new process is spawned to handle requests. The default is 200 threads.

-s maxsegsize
Sets the maximum block size that can be negotiated by the TFTP block size option. The default is 8192.

-f file Specifies a cache file. You can specify a file containing information on files to be preloaded and cached for transmission. A cache file consists of one or more entries. For clarity, place each entry on a separate line. An entry has the form:

a | b <pathname>

where:

- *a* indicates that the specified file is cached in ASCII form. The file is preconverted to netascii format.
- *b* indicates that the specified file is cached in binary form, with no conversion.

Following are examples of cache file entries,

```
a  /usr/local/textfile
b  local/binaryfile
```

If a relative pathname to the file is specified, the TFTP server searches the specified directories for the file.

The cached version of a file is only used for requests requiring the specified format. For example, the binary cached version of a file is not used in satisfying a request for the file in netascii format. If a file is to be retrieved in both binary and ASCII formats, the user must specify that two copies of the file be cached with one in binary format, and the other in netascii format.

Caching is not dynamic. The cache files are read in when the TFTP server is started and are not updated, even if the file on disk is updated. To update or refresh the cache, the TFTP server must be recycled.

-a archive directory
Specifies an archive directory. The files in this directory and its subdirectories are treated as binary files for downloading. This option is useful on EBCDIC machines that act as file servers for ASCII clients. Multiple -a options can be specified; one directory per -a option. Directories must be specified as absolute path names. You can specify no more than 20 directories.

directory
Specifies an absolute path name for a directory. You may specify no more than 20 directories on the tftpd command line.

If the TFTP server is started without a list of directories, all mounted directories are considered active.

If a list of directories is specified, only those directories specified are active. That list is used as a search path for incoming requests specifying a relative path name for a file.

Activating a directory activates all of its subdirectories.

For a file to be readable by the TFTP server, the file must be in an active directory and have world (″other″) read access enabled. For a file to be writable by the TFTP server, the file must already exist in an active directory and have world (″other″) write access.

The TFTP server preforks a child process to handle incoming requests when the concurrency limit is exceeded. Consequently, immediately after starting the TFTP server, two TFTP processes exist.

In case of a flood of concurrent TFTP commands, the TFTP server may fork additional processes. When the number of concurrent requests being processed drops below the concurrency limit, the number of TFTP processes is decreased back to two.

To terminate the TFTP server, send a SIGTERM signal to the oldest existing TFTP process. This is the process with a parent process ID of 1. Termination of this process will cause all of its children to terminate.

Verification of FTP

Verify server

If FTP is in the autolog list and the TCP/IP address space is restarted, FTP should start automatically. For other cases, it should be started manually. To do this, go to the MVS Console and enter the following command:

```
S FTPD
```

Note: The above command assumes the FTP procedure name is FTPD.

If the FTP server startup is complete, the following message should be seen on the MVS console:

```
EZY2702I Server-FTP: Initialization completed at 17:37:29 on  12/17/99.
```

If the message is not seen, a message explaining why FTP did not start up will appear in SYSLOG. Even if the above message is issued, it would be beneficial to inspect SYSLOG for warning messages issued during FTP initialization. EZY2700I displays the port FTP uses as the control port, the port it listens to for incoming connections from clients. In this example, FTP is listening to standard port 21.

The file syslog uses is defined in /etc/syslog.conf. The statement **daemon.info /tmp/daemon.log** directs SYSLOGD to save all the daemon messages in /tmp/daemon.log. Below is an example of output error messages.

```
EZYFT18I Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
EZY2697I IBM FTP CS V1R4 17:34:04 on 10/15/01
EZY2640I Using dd:SYSFTPD=USER1.FTP.DATA for local site configuration parameters
EZYFT46E Error in dd:SYSFTPD file: line 4 near column 9.
EZY2636E SMFLOGN value not specified.
EZYFT46E Error in dd:SYSFTPD file: line 5 near column 8.
EZY2636E SMFREN value not specified.
EZYFT47I dd:SYSFTPD file, line 21: Ignoring keyword "EXTENSIONS REST_STREAM".
EZYFT47I dd:SYSFTPD file, line 29: Ignoring keyword "CTRLCONN".
EZYFT21I Using catalog '/usr/lib/nls/msg/C/ftpdrply.cat' for FTP replies.
EZYFT26I Using 7-bit conversion derived from 'ISO8859-1' and 'IBM-1047' for the control connection.
EZYFT33I Unable to open DDNAME 'SYSFTSX' for the data connection: EDC5129I No such file or directory.
EZYFT31I Using //'TPOUSER.STANDARD.TCPXLBIN' for FTP translation tables for the data connection.
EZYFT09I system information for VIC135: OS/390 version 03 release 12.00 (4381)
EZY2700I Using port FTP control (21)
EZY2701I Inactivity time is 0
EZYFT57I FTP registering with WLM as group = ftpgroup host = VIC135
EZY2702I Server-FTP: Initialization completed at 17:35:05 on 10/15/01.
EZYFT41I Server-FTP: process id 16777255, server job name FTPD11
```

Verify client

To verify that the FTP client works correctly, log onto TSO and issue the NETSTAT HOME command, or issue NETSTAT –h from the z/OS UNIX shell. These commands will show the interface addresses that are known to the system. Below is an example of the output from NETSTAT HOME:

```
netstat home
 MVS TCP/IP NETSTAT CS V1R4       TCPIP NAME: TCPCS           21:10:56
 Home address list:
 LinkName:  OSAQDIO5L
   Address: 9.67.115.13
     Flags: Primary
 LinkName:  LOOPBACK
   Address: 127.0.0.1
     Flags:
 IntfName:  OSAQDIO56
   Address: fec9:c2d4::9:67:115:13
     Type:  Site_Local
     Flags:
 IntfName:  OSAQDIO56
   Address: fec9:c2d4::67:115:13:1234
     Type:  Site_Local
     Flags:
 IntfName:  LOOPBACK6
   Address: ::1
     Type:  Global
     Flags:
 IntfName:  OSAQDIO56
   Address: fe80::6:29dc:21bc:4
     Type:  Link_Local
     Flags:
```

To invoke the FTP client, use any address shown on the NETSTAT HOME address list. The first example below shows how you could log in to the FTP server at 9.67.115.13 using a batch job (the output of the batch job is not shown). The second example shows logging in to the FTP server at 9.67.113.37 from the TSO environment.

```
//FTPBATCH JOB FTPUSER,
// USER=USER1,PASSWORD=TCPSUP
//BATCH  EXEC PGM=FTP
//OUTPUT  DD SYSOUT=*
//INPUT DD *
  9.67.115.13
  USER10 tcpusr
  SITE FILE=SEQ
  QUIT
//*


IBM FTP CS V1R4
FTP: using TCPCS
Connecting to:   9.67.113.37 port: 21.
220-FTPD1 IBM FTP CS V1R4 at vic135, 20:01:42 on 2001-11-07.
220 Connection will close if idle for more than 5 minutes.
>>> FEAT
211- Extensions supported
 SIZE
 MDTM
 REST STREAM
 UTF8
 LANG en*
211 End
>>> LANG en
200 - Language is en-US (United States English)
NAME (9.67.113.37:USER10):

user10
 >>> USER user10
 331 Send password please.
 PASSWORD:


 >>> PASS
 230 USER10 is logged on.  Working directory is "/tmp".
 Command:
```

Verify FTP.DATA statements

Many FTP.DATA statements can be verified via the FTP client STAT and LOCSTAT commands. The output from each installation's STAT and LOCSTAT will depend on the client and server copy of FTP.DATA. Below is sample output of one system.

```
stat
 EZA1701I >>> STAT
 211-Server FTP talking to host 127.0.0.1, port 1027
 211-User: USER1  Working directory: USER1.
 211-The control connection has transferred 2006 bytes
 211-There is no current data connection.
 211-The next data connection will be actively opened
 211-to host 127.0.0.1, port 1027,
 211-using Mode Stream, Structure File, type ASCII, byte-size 8
 211-Automatic recall of migrated data sets.
 211-Automatic mount of direct access volumes.
 211-Auto tape mount is allowed.
 211-Inactivity timer is set to 600
 211-VCOUNT is 59
 211-ASA control characters in ASA files opened for text processing
 211-will be transferred as ASA control characters.
 211-Trailing blanks are removed from a fixed format
 211-data set when it is retrieved.
 211-Data set mode.  (Do not treat each qualifier as a directory.)
 211-ISPFSTATS is set to FALSE
 211-Primary allocation 5 tracks.  Secondary allocation 2 tracks.
 211-Partitioned data sets will be created with 15 directory blocks.
```

```
                          211-FileType SEQ (Sequential - default).
                          211-Number of access method buffers is 5
                          211-RDWs from variable format data sets are discarded.
                          211-Records on input tape are unspecified format
                          211-SITE DB2 subsystem name is DB2
                          211-Data not wrapped into next record.
                          211-Tape write is not allowed to use BSAM I/O
                          211-Truncated records will not be treated as an error
                          211-JESLRECL is 80
                          211-JESRECFM is Fixed
                          211-JESINTERFACELEVEL is 2
                          211-ENcoding is set to SBCS
                          211-SBSUB is set to FALSE
                          211-SBSUBCHAR is set to SPACE
                          211-SMS is active.
                          211-Mgmtclass for new data sets is TCPMGMT
                          211-New data sets will be catalogued if a store operation ends abnormally
                          211-Single quotes will override the current working directory.
                          211-UMASK value is 027
                          211-Process id is 12
                          211-Checkpoint interval is 0
                          211-Authentication type: None
                          211-Record format VB, Lrecl: 128, Blocksize: 6144
                          211 *** end of status ***

                         locstat
                          EZA1600I Trace: FALSE, Send Port: TRUE
                          EZA1601I Send Site with Put command: TRUE
                          EZA2676I Connected to:127.0.0.1, Port: FTP control (21), logged in
                          EZA1605I Local Port: 1027
                          EZA1606I Data type:a, Transfer mode:s, Structure:f
                          EZA2098I Automatic recall of migrated data sets.
                          EZA2100I Automatic mount of direct access volumes.
                          EZA2101I Data set mode. (Do not treat each qualifier as a directory.)
                          EZA2844I ISPFSTATS is set to FALSE
                          EZA2134I Primary allocation 5 tracks, Secondary allocation 2 tracks.
                          EZA2138I Partitioned data sets will be created with 15 directory blocks
                          EZA2103I FileType is SEQ (Sequential - the default).
                          EZA2141I Number of access method buffers is 5.
                          EZA2948I ENcoding is set to SBCS
                          EZA2943I SBSUB is set to FALSE
                          EZA2944I SBSUBCHAR is set to SPACE
                          EZA2142I Mgmtclass for new data sets is TCPMGMT
                          EZA2145I RDW's from VB/VBS files are discarded.
                          EZA2518I Records on input tape are unspecified format
                          EZA2148I DB2 subsystem name is DB2
                          EZA2152I Volid of Migrated Data Sets is MIGRAT
                          EZA2154I Trailing blanks in records read from RECFM F datasets are discarded.
                          EZA2535I Record format: VB, Lrecl: 128, Blocksize: 6144.
                          EZA2801I Data not wrapped into next record.
                          EZA2529I Truncated records will not be treated as an error.
                          EZA2494I Checkpoint interval is 0
                          EZA2511I Checkpoint data set will be opened for GET
                          EZA2428I CHKPTPrefix uses Home to determine the HLQ of the FTP.CHECKPOINT file.
                          EZA2817I Automatic mount of tape volumes.
                          EZA2809I CCONNTIME is 120
                          EZA2810I DATACTTIME is 120
                          EZA2811I DCONNTIME is 120
                          EZA2812I INACTTIME is 120
                          EZA2813I MYOPENTIME is 120
                          EZA2815I VCOUNT is 59
                          EZA2689I Prompting: ON, Globbing: ON
                          EZA2719I ASA control characters transferred as ASA control characters
                          EZA2720I New data sets catalogued if a store operation terminates abnormally
                          EZA2722I Single quotes will override the current working directory
                          EZA2724I UMASK value is 027
```

```
EZA2819I Data connections for the client are not firewall friendly.
EZA2889I Authentication mechanism: None
EZA2866I Tape write is not allowed to use BSAM I/O
EZY2640I Using 'SYS1.TCPPARMS(FTPDATA)' for local site configuration parameters.
EZA1460I Command:
```

Verifying anonymous, banner, and other optional configuration information

Depending on your installation's choices for anonymous level, banner support chosen, exits, and so on, verification of support output will differ. To verify anonymous configuration at a particular installation, log in as anonymous and verify the behavior is as expected. For example, if EMAILADDRCHECK FAIL is specified in FTP.DATA, try to log in as anonymous using an incorrect e-mail address as password. To verify banner support, login and verify the banners are displayed as expected. Below is a sample of FTP.DATA and FTP client output for one such installation.

```
; BANNER STUFF
EMAILADDRCHECK FAIL
BANNER USER1.TEST1
ADMINEMAILADDR FTPADMIN@MYSYSTEM.COM
; ANONYMOUS STUFF
ANONYMOUSLEVEL 3
STARTDIRECTORY HFS
```

```
ftp 9.67.113.63
 IBM FTP CS V2R10  1999 349 01:35 UTC
 FTP: using TCPCS
 Connecting to:  9.67.113.63 port: 21.
 220-FTPD1 IBM FTP CS V2R10 at HOSTA, 19:02:45 on 1999-12-17.
 220-You have just read 'USER1.TEST1'
 220-ADMINEMAILADDRESS is FTPADMIN@MYSYSTEM.COM
 220 Connection will not timeout.
 NAME (9.67.113.63:USER4):
anonymous no-email-pw
 >>> USER anonymous
 331 Send password please.
 >>> PASS
 530 PASS command failed.
 Command:
```

Verify FTP-JES interface (optional)

As with the other optional configuration information, FTP-JES support can best be verified by logging in and confirming the FTP.DATA parameters chosen. To verify JES support, a simple batch job can be created if the JESINTERFACELEVEL is set to the security requirements of an installation. Below is the batch job and FTP client output for JESINTERFACELEVEL 2.

```
EDIT       USER1.FTP.JCL.TEST                              Columns 00001 00072
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ****************************
000100 //JOBTEST  JOB MSGCLASS=H,MSGLEVEL=(1,1),CLASS=A,
000200 //         USER=USER1
000300 //STEP1    EXEC PGM=IEBGENER
000400 //OBJTMP1  DD DSN=&PRLOBJ,DISP=(NEW,PASS,DELETE),
000500 //            SPACE=(CYL,(1,1,10)),
000600 //            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
000700 //SYSPRINT DD SYSOUT=A
000800 //SYSUT1   DD DSN=SYS1.PROCLIB(JES2),DISP=SHR
000900 //SYSIN    DD DUMMY
001000 //SYSUT2   DD SYSOUT=H
001100 //
001200 //         EXEC PGM=IEFBR14
****** **************************** Bottom of Data ***************************
```

```
site file=jes jesjobname=jobtest jesowner=* jesstatus=all
 EZA1701I >>> SITE file=jes jesjobname=jobtest jesowner=* jesstatus=all
 200 Site command was accepted
 EZA1460I Command:
put 'user1.ftp.jcl.test'
 EZA1701I >>> SITE FIXrecfm 80 LRECL=80 RECFM=FB BLKSIZE=32720
 200 Site command was accepted
 EZA1701I >>> PORT 127,0,0,1,4,12
 200 Port request OK.
 EZA1701I >>> STOR 'user1.ftp.jcl.test'
 125 Sending Job to JES internal reader FIXrecfm 80
 250-It is known to JES as JOB00076
 250 Transfer completed successfully.
 EZA1617I 984 bytes transferred in 0.005 seconds.  Transfer rate 196.80 Kbytes/sec.
 EZA1460I Command:
dir j76
 EZA1701I >>> PORT 127,0,0,1,4,13
 200 Port request OK.
 EZA1701I >>> LIST j76
 125 List started OK for JESJOBNAME=JOBTEST, JESSTATUS=ALL and JESOWNER=*
 EZA2284I JOBNAME  JOBID    OWNER    STATUS CLASS
 EZA2284I JOBTEST  JOB00076 USER1    OUTPUT A        RC=000
 EZA2284I          ID  STEPNAME PROCSTEP C DDNAME   BYTE-COUNT
 EZA2284I          001 JESE              H JESMSGLG      1084
 EZA2284I          002 JESE              H JESJCL        1023
 EZA2284I          003 JESE              H JESYSMSG      1143
 EZA2284I          004 STEP1             H SYSUT2         741
 EZA2284I          005 STEP1             A SYSPRINT       209
 EZA2284I 5 spool files
 250 List completed successfully.
 EZA1460I Command:
```

Chapter 10. Domain Name System (DNS)

This chapter contains information about configuring the name server in a BIND-based Domain Name System (DNS). BIND was developed at the University of California, Berkeley and is currently maintained by the Internet Software Consortium (ISC). The name servers discussed in this chapter are based on BIND 4.9.3 and BIND 9.

This chapter also contains information about connection optimization, which uses DNS for distributing connections among hosts or server applications within a sysplex domain.

The Domain Name System is a client/server model in which programs called *name servers* contain information about host systems and IP addresses. Name servers provide this information to clients called *resolvers*.

This chapter is not intended to be a comprehensive description of DNS or of BIND. For more complete descriptions, refer to the latest edition of *DNS and BIND* by Paul Albitz and Cricket Liu (O'Reilly & Associates, Inc.).

DNS and BIND overview

TCP/IP applications map fully qualified domain names to 32-bit IPv4 IP addresses or 128-bit IPv6 addresses to identify network nodes. The z/OS BIND 9 name server supports resource records for IPv6 address mapping. It also accepts IPv6 connections, depending on the z/OS TCP/IP stack setup and profile, and on the name server configuration. While TCP/IP applications refer to host computers by their IP addresses, it is easier to use host names. To enable the use of host names in a network, the Domain Name System (DNS) translates host names to IP addresses. Mapping must be consistent across the network to ensure interoperability. DNS provides the host name-to-IP address mapping through network server hosts called *domain name servers*. For detailed information about name servers, see "Domain name servers" on page 419. DNS can also provide other information about server hosts and networks such as the TCP/IP services available at a server host and the location of domain name servers in a network.

DNS organizes the hosts in a network into domains. A *domain* is a group of hosts that share the same name space in the domain hierarchy and are usually controlled within the same organization. Domains are arranged in a hierarchy. A special domain known as the *root domain* exists at the top of the hierarchy. The root domain servers store information about server hosts in the root domain and the name servers in the delegated, *top-level* domains, such as *com* (commercial), *edu* (education), and *mil* (military). The name servers in the top-level domain, in turn, store the names of name servers for their delegated domains, and so on.

The complete name of a host, also known as the *fully qualified domain name* (FQDN), is a series of labels separated by dots or periods. Each label represents an increasingly higher domain level within a network. The complete name of a host connected to one of the larger networks generally has more than one subdomain, as shown in the following examples:

```
host1.subdomain2a.subdomain2.rootdomain
user4720.eng.mit.edu
```

A domain name server requires the FQDN. The client resolver combines the host name with the domain name to create the FQDN before sending the name resolution request to the domain name server.

DNS also provides IP address-to-host name mapping. The DNS defines a special domain called *in-addr.arpa* to translate IPv4 addresses to host names, and the *ip6.int* and *ip6.arpa* domains for IPv6 address-to-host name translation. This kind of mapping is useful for producing output (host names) that is easy to read. An in-addr.arpa name is composed of the reverse octet order of an IP address concatenated with the *in-addr.arpa* string. For example, a host named `Host1` has `9.67.43.100` as an IP address. The in-addr.arpa domain translates the `Host1` IP address `9.67.43.100` to `100.43.67.9.in-addr.arpa`.

For IPv6 reverse lookups, BIND 9 supports the *bitstring* and *nibble* formats.

A system administrator can name the host systems and domains in the local, private network with any name you want, but to link with name servers in a public network like the Internet, you need to determine which domain you want to be in (which parent domain) and then contact the registrar in that domain to register the names and IP addresses of your name servers. This ensures that queries from outside the domain being defined can be answered by this name server if need be.

Note: Contact the InterNetwork Information Center (InterNIC) for more information about Internet registration. You can contact InterNIC by pointing your Web browser at http://www.internic.net.

Domain names

The DNS uses a hierarchical naming convention for naming hosts. Each host name is composed of domain labels separated by periods. Local network administrators have the authority to name local domains within an intranet. Each label represents an increasingly higher domain level within an intranet. The fully qualified domain name of a host connected to one of the larger intranets generally has one or more subdomains:

- *host.subdomain.subdomain.rootdomain*
- *host.subdomain.rootdomain*

Domain names often reflect the hierarchy level used by network administrators to assign domain names. For example, the domain name `eng.mit.edu.` is the fully qualified domain name, where `eng` is the host, `mit` is the subdomain, and `edu` is the highest level domain (root domain).

Figure 57 on page 419 is an example of the DNS used in the hierarchy naming structure across an intranet.

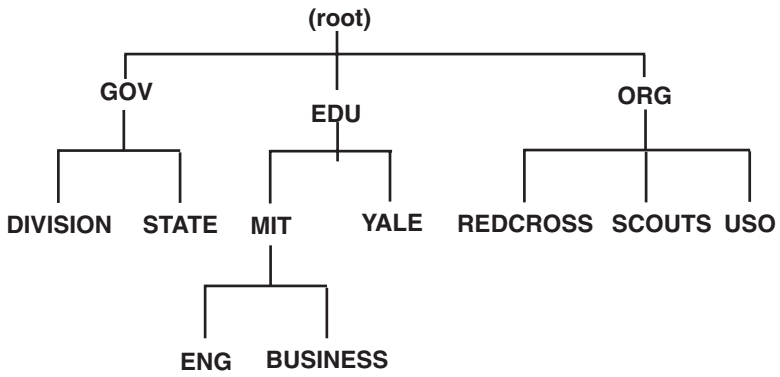


Figure 57. Hierarchical naming tree. Hierarchical naming tree

You can refer to hosts in your domain by host name only; however, a name server requires a fully qualified domain name. The local resolver appends the domain name before sending the query to the Domain Name Server for address resolution.

Domain name servers

Domain name servers are designated network nodes that maintain a database of information about all nodes in some part of the domain name space, called a *zone*. A name server is said to be *authoritative* for its zone. A zone consists of the resources within a single domain (for example, commercial or `.com`) or subdomain (for example, `raleigh.ibm.com`). Typically, a zone is administered by a single organization or individual. The complete database is not kept by any one name server on a network. A name server is authoritative only within its zone of authority.

All host systems in a given zone share the same higher level domain name (for example, `host1.raleigh.ibm.com`, `host2.raleigh.ibm.com`, `host3.raleigh.ibm.com`, and so on). As system administrator, you create a zone of authority by listing all the host systems in your zone in the database file of the name server that is authoritative for the zone.

If a domain name server receives a query about a host for which it has information in its database or in its cache, it performs the name resolution and returns all the address records associated with the host to the client. Some hosts (for example, routers or gateways between two or more networks) might have more than one IP address.

Alternatively, the name server can query other name servers for information. This process is called *iterative resolution*. The local name server successively queries other name servers, each of which responds by referring the local name server to a remote name server that is closer to the name server authoritative for the target domain. Finally, the local name server queries the authoritative name server and gets an answer. If the information about a requested host name does not exist or if a name server does not know where to go for the information, it sends a negative response back to the client.

There are multiple name server modes in the DNS:

- Authoritative
  - Master (primary)
  - Slave (secondary)
- Caching-only servers

- Forwarders
- Stealth

A single server can perform multiple functions. For example, it can be a primary server and a slave server for different zones. The purpose of having these different kinds of servers is to provide redundancy (in case of system failure), to distribute the workload among multiple servers, to speed up the name-resolution process, and to provide flexibility in network design. In addition to being an authoritative or caching-only server, a name server can be defined to only contact a specific set of name servers if queries cannot be resolved locally (through the use of forwarders).

The following sections discuss authoritative servers, caching-only servers, and forwarding.

Authoritative servers

An authoritative server is the authority for its zone. It queries and is queried by other name servers in the DNS. The data it receives in response from other name servers is cached. Authoritative servers are not authoritative for cached data.

There are two types of authoritative servers: master (primary) and slave (secondary). Each zone must have only one master name server, and it should have at least one slave name server for backup to minimize dependency on a particular node. Calling a *particular* name server a master or slave is misleading. Any given name server can take on either or both roles, as defined by the boot or conf file.

The zone data updates and maintenance are reflected in the master name server. The slave name servers update their databases by contacting the master name server at regular intervals or possibly (BIND 9) after being notified of an update by the master name server. Both master and slave name servers are authoritative for a zone.

The zones of authority are arranged in a hierarchy based on the domain origin components. A special zone known as the *root* exists at the top of the domain name hierarchy in a network. The root zone contains a list of all the root servers. For example (see Figure 57 on page 419), in the Internet, the root name servers store information about nodes in the root domain, and information about the delegated domains, such as `com` (commercial), `edu` (education), and `mil` (military). The root name servers store the names of name servers for each of these domains, which in turn store the names of name servers for their delegated subdomains.

TCP/IP applications contact a name server whenever it is necessary to translate a domain name into an IP address, or when information is required about a domain. The name server performs the translation if it has the necessary information. If it does not have the necessary information, the name server can contact other name servers, which in turn can contact other name servers. This process is called a *recursive query*. Alternatively, a name server can simply return the address of another name server that might hold the requested information. This is called a *referral response* to a query. Name server implementations must support referrals, but are not required to perform recursive queries. See “Resolvers” on page 422 for more information about query responses.

Master name servers: A master name server maintains all the data for its zone. Static resources are kept in database files called *domain data files*. For information on creating domain data files, see “Step 5. Create the domain data files (master name server only)” on page 433. Master name servers can also receive zone

updates dynamically. For information on dynamic DNS, see “Dynamic IP” on page 496. For information on dynamic generation of resources, see “Connection optimization in a sysplex domain” on page 482.

Slave name servers: A slave name server acts as an alternate to the master server if the master name server becomes unavailable or overloaded. The slave name server receives zone data directly from the master name server in a process called *zone transfer*. Zone transfers, which only occur when data has changed, are based on the refresh interval in the Start of Authority (SOA) resource record or, for BIND 9 name servers only, on using the DNS Notify function. For a description of the SOA resource record, see *z/OS Communications Server: IP Configuration Reference*. A slave server, like a master server, is authoritative for a zone.

Caching-only servers

All name servers cache (store) the data they receive in response to a query. A caching-only server, however, is not authoritative for any domain. Responses derived from cached information are flagged in the response. When a caching-only server receives a query, it checks its cache for the requested information. If it does not have the information, it queries a local name server or a root name server, passes the information to the client, and caches the answer for future queries. The names and addresses of the root name servers are acquired from the servers listed in the hints file, the name and file path of which are specified in the name server's configuration file.

You can use caching servers to create a large cache of responses to frequently requested queries and reduce the number of queries made to master servers. The caching server stores data for a period of time determined by the time-to-live (ttl) value, and the cached information is lost if the name server is restarted.

Forwarders

Normally, name servers answer queries from cached data or, if that does not succeed, they attempt to contact other name servers identified in their data files as authoritative for certain domains. However, name servers can also be configured to contact special servers called *forwarders* before contacting the name servers listed in their data files. If a forwarder cannot process the query and if the local name server is not a forward-only name server, the local name server contacts the name servers in its data files. A forward-only name server relies completely on its forwarders. It does not try to contact other servers to find out information if the forwarders do not give it an answer.

The forwarding function is useful for reducing the number of queries to servers on the Internet and for creating a large cache of information on forwarders. It is also a useful function for providing Internet access for local servers that, for one reason or another, do not have access themselves.

Stealth server

A *stealth server* is a server that answers authoritatively for a zone, but is not listed in that zone's NS records. Stealth servers can be used as a way to centralize distribution of a zone, without having to edit the zone on a remote nameserver. When the master file for a zone resides on a stealth server in this way, it is often referred to as a *hidden primary* configuration. Stealth servers can also be a way to keep a local copy of a zone for rapid access to the zone's records, even if all official nameservers for the zone are inaccessible.

Resolvers

Programs that query a name server are called *resolvers*. Because many TCP/IP applications need to query the name server, a set of routines is usually provided for application programmers to perform queries. On z/OS, these routines are available in the resolver provided by z/OS Communications Server.

z/OS CS provides programs for interactively querying a name server:

- NSLOOKUP (TSO)
- onslookup/nslookup (z/OS UNIX)
- DIG (TSO)
- dig (z/OS UNIX)
- host

Note: The nsupdate program also makes queries to name servers as part of its operations.

For information on these programs, see *z/OS Communications Server: IP System Administrator's Commands*.

The BIND 4 onslookup and TSO DIG commands use the resolver provided by z/OS Communications Server for all their resolver facilities. The BIND 9 onslookup and dig commands use the resolver initialization facilities of the resolver provided by z/OS Communications Server, but use their own resolver for any additional resolver facilities needed.

Resolver directives for nslookup

The onslookup program uses the following resolver directives (TCPIP.DATA statements):

- domain/domainorigin
- search
- nameserver/nsinteraddr
- sortlist
- options debug/options ndots

Resolver directives for dig

The dig program uses the following resolver directives (TCPIP.DATA statements):

- domain/domainorigin
- search
- nameserver/nsinteraddr
- options ndots

Query Packets

Resolvers operate by sending query packets to a name server, either over the network or to the local name server.

A query packet contains the following fields:

- Domain name
- Query type
- A query class

For information on valid query class (network class) and query type (data type) values, see *z/OS Communications Server: IP Configuration Reference*. The name

server attempts to match the three fields of the query packet to its database. For flexibility, the following wildcard query types are defined:

Type	Description
ANY	Indicates any record type for the domain name.
AXFR	Indicates the query type used by secondary name servers to transfer all records in the zone. (The query class is set to IN when using the AXFR query type.)
MAILB	Indicates any mailbox records for the domain name.

The name server can return the following query responses:

Response	Description
Authoritative	Is returned from a primary or secondary name server. The name server contains all the domain data used to define the zone for the specified query.
BADVERS	The name server received a request which contained a bad EDNS version.
Nonauthoritative	Is returned from a cache kept by a name server. The cache does not contain the domain data used to define the zone for the specified query.
Format Error	The name server found an error in the query packet sent by the resolver.
Name Error	No resource records of any type (including wildcards) exist for the domain name specified.
NXDOMAIN (negative)	No records of the requested type were found for the domain name specified.
Not-implemented	The name server does not support the type of query requested.
NOTAUTH	The name server is not authoritative for the zone.
NOTZONE	A dynamic update failed because the name to be updated is not contained within the given zone.
NXRRSET	A dynamic update failed because the prerequisites were not satisfied. The Resource Record set existed when the prerequiste stated it should not.
Referral	Contains the addresses of other name servers that might be able to answer the query. A referral response is returned when a recursive query is not supported, not requested, or cannot be answered because of network connectivity.
Refused	The name server refuses to perform the specified operation. For example, some root name servers limit zone transfers to a set number of IP addresses.
YXDOMAIN	DNAME mapping failed because the new name was too long.
YXRRSET	A dynamic update failed because the prerequisites were not satisfied. The Resource Record set did not exist when the prerequiste stated it should.

Resource Records

Data from a name server is stored and distributed in a format known as a resource record. Resource record fields are described in detail in *z/OS Communications Server: IP Configuration Reference*. Each response from a name server can contain

several resource records, which can contain a variety of information. The format of a response is defined in RFC 1035. It includes the following sections:

- A question section, echoing the query for which the response is returned.
- An answer section, containing resource records matching the query.
- An additional section, containing resource records that do not match the query, but might provide useful information for the client. For example, the response to a query for the host name of a name server for a specific zone includes the IP address of that name server in the additional section.
- An authority section, containing information specific to the type of response made to the query. If a referral is returned, this section contains the domain names of name servers that could provide an authoritative answer. If a negative response is returned indicating the name does not exist, this section contains a Start Of Authority (SOA) record defining the zone of authority of the responding name server.

Recommended reading

DNS and BIND, 4th Edition by Paul Albitz and Cricket Liu (O'Reilly & Associates, Inc.) gives a comprehensive description of DNS and BIND, and specifically contains information on BIND 9.1.0. The BIND 9 name server in V1R4 is based upon BIND 9.2.0.

For additional information on DNS in a sysplex, refer to the following Redbooks:

- *z/OS eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 2: UNIX Applications, SG24-5228*
- *TCP/IP in a Sysplex, SG24-5235-01*

For information on Dynamic IP, see "Dynamic IP" on page 496.

If you wish to participate, a BIND users mailing list can be subscribed to at `http://www.isc.org/products/BIND/`

DNS protocols are described in various Request for Comments (RFC) papers and Internet drafts. RFCs outline existing protocols, suggest new protocols, and establish standards for the Internet protocol suite. Internet drafts are proposals, techniques, and mechanisms that document Internet Engineering Task Force (IETF) work-in-progress.

For information about obtaining RFCs, refer to `http://ftp.ietf.org/rfc.html`.

A list of RFCs related to DNS are located in "DNS-related RFCs" on page 535.

Migrating to BIND 9

Refer to the *z/OS Communications Server: IP Migration* for detailed information.

Performance issues

The BIND 9 name server supports multithreading and DNSSEC which creates extra overhead. Multithreading might improve performance for large zones but can be a disadvantage for small zones. The BIND 9 name server might perform slower than the BIND 4.9.3 name server for small zones on simple query and response operations, or might be as fast as a BIND 4.9.3 name server but with higher CPU consumption, depending on whether your system is already CPU constrained.

Because of the multithreading, BIND 9 name servers are able to answer queries during zone transfers. BIND 4.9.3 name servers are unable to answer queries for a period of time during zone transfers; the larger the zone, the more noticeable this can become. BIND 9 name servers are also capable of Incremental Zone Transfers, while BIND 4.9.3 name servers are not. Incremental Zone Transfer allows only the changed information in a zone to be sent to slave name servers instead of the entire zone. If your name servers employ dynamic update for frequent zone changes, the Incremental Zone Transfer feature of BIND 9 might offer some performance advantages while reducing network traffic.

The use of DNSSEC (authenticating DNS data with digital signatures) will have a performance cost. The authentication process requires more CPU, and signing a zone greatly increases the zone's size. DNS message sizes will also increase between client and server, and between DNS servers. If the message size becomes too large for UDP, the message will be sent by TCP, which is more resource intensive.

Since the BIND 9 name server is multithreaded, it can take advantage of any additional processors you add to the system. The BIND 9 name server will detect the number of logical CPUs configured for the system (if not running partitioned) or LPAR (if running partitioned), and create additional worker threads accordingly. For relatively simply configured name servers that are small, are not using DNSSEC, or are not kept busy, the overhead in managing the extra threads created on a multiprocessor image can actually be disadvantageous. If you feel this might be the case, you can override the number of worker threads created by using the *-n* option when starting the name server. The number of logical CPUs detected (and therefore, the number of worker threads created by default) is logged when the name server is started.

Compatibility considerations

Zone transfers

It is not recommended to have a BIND 4.9.3– DNS act as a slave name server to a BIND 9 master. However, if required, it can be done under certain conditions. If the BIND 9 master contains any resource records (RRs) that BIND 4.9.3–DNS does not understand, the zone will fail to load. A BIND 9 nameserver with BIND 4 slaves should specify the 'transfer-format one-answer' option in its named.conf. Otherwise, any transfer will also fail.

BIND 4.9.3 does not understand NOTIFY, therefore, if BIND 4.9.3 is running as a slave name server to a BIND 9 master, the DNS Change Notification protocol will not work. The standard method of zone transfers applies, where the slave periodically polls the master for an updated SOA serial number.

Queries

BIND 4.9.3 can participate in DNS queries when BIND 9 name servers are in the DNS tree structure, even when the queries are for RR types that BIND 4.9.3 does not understand. For example, if a resolver is pointed to a BIND 4.9.3–DNS and is asked for an AAAA record (an RR type that BIND 4.9.3 name server does not understand), BIND 4.9.3–DNS will recursively query other (possibly BIND 9) name servers and return the answer to the client. The BIND 4.9.3–DNS will not cache the response (in the case of RR types it does not understand), which may mean a little more network traffic. This caching issue may or may not be significant depending on your particular network traffic patterns.

Dynamic update

BIND 4.9.3 version of Dynamic Update is incompatible with the BIND 9 version. For dynamic update on BIND 4.9.3-DNS, use nsupdate with -V v4 start option. For dynamic update on BIND 9-DNS, use nsupdate with -V v9 start option. Additionally, only the DHCP server on z/OS and OS/2 can successfully dynamically update the BIND 4.9.3–DNS.

DNSSEC

BIND 4.9.3 does not support DNSSEC.

TSIG

BIND 4.9.3 does not support TSIG security which may be used on queries, update and zone transfers on BIND 9 name servers.

DNS/WLM (Sysplex connection balancing)

Cannot be done by a name server running in BIND 9 mode.

IPv6 support

Not supported by a BIND 4.9.3 name server.

Stack affinity

The BIND 9 name server is a generic server, unlike the BIND 4.9.3 name server which has stack affinity. If stack affinity is desired for the BIND 9 name server, use the _BPXK_SETIBMOPT_TRANSPORT environment variable.

NOTIFY

This function notifies the slave of a change in the master. It is not supported by a BIND 4.9.3 name server.

Running the name server in BIND 9 and BIND 4.9.3 mode simultaneously

The following describes the tasks involved in setting up the name server to run in BIND 9 and BIND 4.9.3 modes simultaneously:

1. Bind each name server to its own set of IP interfaces.
   a. Bind the BIND 9 name server to the set of IPv4 interfaces you wish to be serviced by the BIND 9 name server using the 'listen-on{}' option in the named.conf file. The BIND 4.9.3 name server is unable to listen on IPv6 interfaces while the BIND 9 name server can. Therefore, IPv6 interfaces cannot be shared among the BIND 4.9.3 and BIND 9 name servers. To further specify interfaces for queries, transfers and notifies from a BIND 9 name server, the following BIND 9 configuration file options are also available:
      - query-source{}
      - transfer-source{}
      - notify-source{}

   b. The BIND 4.9.3 name server cannot specify its own IPv4 interfaces in its configuration file but it will listen, query and transfer on the IPv4 interfaces not taken by the BIND 9 name server.

Both servers can share port 53 if they connect to clients or other servers on different IP addresses.

2. Specify port ownership.

   Assign unique job names to the BIND 4.9.3 and BIND 9 name servers. Reserve port 53 for TCP for both job names. Port 53 TCP port reservation jobname must have a suffix of '2' for BIND 4.9.3 and a suffix of '1' for BIND 9. UDP port reservation for multiple job names is not allowed.

3. Use the _BPX_JOBNAME environment variable if starting the name server from the z/OS UNIX shell to distinguish the job names of the two name server daemons.

   Notes:

   a. MVS jobname cannot be *named* for both BIND 4 and BIND 9 servers (for example, call them namedv4 and namedv9).

   b. PORT 53 UDP can only be reserved for one jobname because of a TCP/IP profile restriction.

   c. Jobname/step is unpredictable if the name server is directly started from the z/OS UNIX shell.

   d. Whether started from an MVS procedure or the z/OS UNIX shell, the port can be generically reserved to UNIX applications: PORT 53 TCP (also UDP) OMVS.

4. Store the name server process IDs (PIDs) in unique files.

   Configure the BIND 9 name server to store the process ID (PID) in a file other than the one used for the BIND 4.9.3 name server (/etc/named.pid). This is done with the 'pid-file' named.conf file option.

5. Configure client resolvers.

   Configure clients' resolver configuration data set or HFS file so that it points to the interface or interfaces of the desired name server. This is typically specified by the NSINTERADDR statement, or an equivalent statement. For more information on how to configure the resolver, refer to "Understanding resolvers" on page 12.

Setting up and running the name server

This section describes the tasks involved in configuring the name server and verifying that the name server is working correctly.

Name server configuration files are arranged in a Hierarchical File System (HFS). Before configuring DNS, the TSO user ID from which the name server is started must have the proper authority to access the name server configuration and zone files. For a complete description of file permissions within the HFS, refer to *z/OS UNIX System Services Planning*.

Configuring a master (primary) name server

The name resolution process is an example of a client/server relationship in which clients, through their resolvers, request a service (name resolution) from name servers. For a general overview of name servers, see "Domain name servers" on page 419.

The following summary lists the steps for configuring a master server or a caching-only server:

1. Create a configuration file for your environment. Select a) or b):
   a. Create the boot file for BIND 4.9.3–DNS .

b. Create the configuration file for BIND 9–DNS.
2. For BIND 4.9.3 DNS only — Specify stack affinity (multiple stack environment).
3. Specify port ownership.
4. Update the name server start procedure.
5. Create the domain data files (master name server only).
6. Create the hints (root server) file.
7. Create the loopback file.
8. Ensure that the syslog daemon is running on your system.
9. Specify whether the name server is to run swappable or nonswappable.
10. Start the name server.
11. Verify that the name server started correctly.
12. Verify that the name server can accept queries.

The difference between configuring a master (primary) name server and slave (secondary) and caching-only servers is the creation of domain data files (the database files containing host-to-address and address-to-host mappings). The domain data files are maintained on the master name server, and the slave name server transfers this data to its own database. Examples of slave, caching-only, and forward-only configurations are in “Configuring a slave name server” on page 450, “Configuring a cache-only name server” on page 453, and “Adding forwarding to your name server” on page 456.

Note: Continue with selecting Step 1a. *or* 1b.

Step 1a. Create the boot file for BIND 4.9.3–DNS

The boot file is the main configuration file for a domain name server. The named daemon reads the boot file for information about how to set up the local name server. The records in the boot file identify the type of name server, the zones over which it has authority, the location of data for setting up its name resolution database, and other configuration options. The default name of the boot file is `/etc/named.boot`. You can specify an alternate boot file using the `-b` named start option. For information about named options, refer to *z/OS Communications Server: IP Configuration Reference*.

Note: The named daemon reads the boot file only when the named daemon starts or when it receives a SIGHUP signal. For a description of named signals, refer to *z/OS Communications Server: IP Configuration Reference*.

Each type of name server has a special boot file configuration. You create a boot file using directives. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Boot files created locally for use by the name server are assumed to be in code page IBM-1047. For systems using other code pages, use the `iconv` command to translate from the local code page to code page IBM-1047. See *z/OS UNIX System Services Command Reference* for more information.

A boot file for a master name server (a name server that maintains all the data for its zone in database files) will need, at a minimum, to specify the zones for which the name server will be authoritative, their locations in the HFS, and the location of the hints file (the location of root name servers). A loopback file is also recommended.

This example illustrates a name server acting as master for the forward zone mycorp.com and the reverse zone 34.37.9.in-addr.arpa.

The sample BIND 4.9.3–DNS boot file shipped in /usr/lpp/tcpip/samples/named.boot is shown below. Refer to the program directory for its location.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;  Note:  This file must be copied and renamed to /etc/named.boot and all
;         zone files referenced below must be copied to /etc/dnsdata/ for
;         this file to function as intended.
;
;  /etc/named.boot
;
;      boot file for name server
;
;
;type    domain                  source file or host
;
directory /etc/dnsdata     {1}
primary  mycorp.com   {2}        db.mycorp.v4        {3}
primary  34.37.9.in-addr.arpa    db.34.37.9.v4   {4} {5}
primary  0.0.127.in-addr.arpa    db.loopback.v4      {7}
cache  .                        db.cache             {6}
options query-log     {8}
```

This boot file specifies:

1. The location of the files (/etc/dnsdata).
2. The name server will be the primary name server for the mycorp.com zone.
3. The data for mycorp.com is contained in db.mycorp.v4.
4. The name server will be the primary name server for the reverse mapping zone, 34.37.9.in-addr.arpa.
5. The data for addresses 9.37.34.x contained in the zone will be specified in db.34.37.9.v4.
6. The list of root name servers is in db.cache.
7. The name server defines the loopback address in the 0.0.127.in-addr.arpa zone and the data is contained in the file, db.loopback.v4.
8. All queries coming in to this name server will be logged in the syslog daemon output file.

Step 1b. Create the configuration file for BIND 9–DNS.

The sample BIND 9-DNS configuration file shipped in /usr/lpp/tcpip/samples/named.conf is shown below. Refer to the program directory for its location.

```
#           LICENSED MATERIALS - PROPERTY OF IBM
#             "RESTRICTED MATERIALS OF IBM"
#          5694-A01 (C) COPYRIGHT IBM CORP. 2001
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 1993
# All Rights Reserved
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# Licensed Materials - Property of IBM
#
#
#       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
#
# INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
# EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
# WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
# LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
# OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
# IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
# YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
# CORRECTION.
#
# Note:  This file must be copied and renamed to /etc/named.conf and all
#        all zone files referenced below must be copied to /etc/dnsdata/ for
#        this file to function as intended.  In addition, the default location
#        for the process id file is in /var/run/pid.file; if that directory
#        does not exist a different one can be configured with the option:
#
#        pid-file "path/file-name";
#
#  /etc/named.conf
#
#        conf file for name server
#

options {
   directory "/etc/dnsdata";
};

logging {
   category "queries" {
      default_syslog;
   };
};

zone "mycorp.com" in {
   type master;
   file "db.mycorp.v9";
};

zone "34.37.9.in-addr.arpa" in {
   type master;
   file "db.34.37.9.v9";
};

zone "0.0.127.in-addr.arpa" in {
   type master;
   file "db.loopback.v9";
};
```

```
zone "." in {
   type hint;
   file "db.cache";
};
```

Step 2. For BIND 4.9.3–DNS only: specify stack affinity (Multiple stack environment)

In a multiple stack environment, the name server is like any application. It binds to the stack specified by TCPIPJOBNAME. To run multiple name servers, each must use a different TCPIPJOBNAME. See “Considerations for multiple instances of TCP/IP” on page 54 for more information about specifying TCPIPJOBNAME.

Note: When changing TCPIPJOBNAME, any client needing to be connected to the new stack name must be restarted (for example, any application, including the name server, that is currently running, that is bound to the new stack name specified by TCPIPJOBNAME).

Step 3. Specify port ownership

The name server uses a single port (53) for TCP and UDP sessions. Both servers can share port 53 if they connect to clients or other servers on different IP addresses:

- A BIND 9 server can specify IP addresses to listen on and from which to send queries, notifies and zone transfers in its configuration file.
- A BIND 4 server cannot specify IP addresses in its configuration file but will use any IP address not already selected by a BIND 9 server.

To specify port ownership when using the named start procedure for BIND 4.9.3, add the following statements to the PROFILE.TCPIP data set:

```
PORT
  53 TCP NAMED2
  53 UDP NAMED2
```

To specify port ownership when using the named start procedure for BIND 9, add the following statements to the PROFILE.TCPIP data set:

```
PORT
  53 TCP NAMED1
  53 UDP NAMED1
```

Notes:

1. MVS jobname cannot be *named* for both BIND 4 and BIND 9 servers (for example, call them namedv4 and namedv9).
2. The jobname on the TCP and UDP port reservation statements requires a suffix of *2* for BIND 4.9.3.
3. The jobname on the TCP and UDP port reservation statements requires a suffix of *1* for BIND 9.
4. PORT 53 UDP can only be reserved for one jobname because of a TCP/IP profile restriction.
5. Jobname/step is unpredictable if the name server is directly started from the z/OS UNIX shell.
6. Whether started from an MVS procedure or the z/OS UNIX shell, the port can be generically reserved to UNIX applications: PORT 53 TCP (also UDP) OMVS.

For more information on the PORT statement, refer to *z/OS Communications Server: IP Configuration Reference*.

Note: In order to pick up changes in the PROFILE.TCPIP data set, stop and restart TCP/IP. As an alternative to stopping the stack, use the VARY TCPIP,,OBEYFILE command to reserve the ports while the stack is up.

Step 4. Update the name server start procedure (Optional)

When choosing to start the name server from MVS, create a start procedure. This is not necessary if the name server is started from the z/OS UNIX shell. Move the sample start procedure, SEZAINST(NAMED), to a recognized PROCLIB. Specify name server parameters and change the data set names as required to suit local configuration. The boot file path (for BIND 4.9.3–DNS) or the conf file path (for BIND 9–DNS) can also be changed as shown below in the sample start procedures. If you want to have NAMED messages written out to SYSLOGD (HFS file) instead of the system console (syslog), then you must start NAMED via BPXBATCH as shown below:

BIND 4.9.3–DNS:

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZANSPRO
//*
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        5694-A01 (C) Copyright IBM Corp. 1997, 2000.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//*        NAMED can be started with a variety of parameters.
//*        In this example, the "-b" parameter describes which
//*        boot file NAMED should be started with.
//*
//NAMED    PROC B='/etc/named.boot'
//NAMED    EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
//         PARM='PGM /usr/lpp/tcpip/sbin/named -b &B '
//*
//*        NAMED can use certain environmental variables, such
//*        as NLSPATH (to determine the location of the message
//*        catalog), and RESOLVER_CONFIG (to determine the location
//*        of the file that contains the parameter TCPIPjobname).
//*        These variables can be specified in a file defined
//*        by STDENV.
//*        An example of the contents of this file follows:
//*
//*             RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*                          or
//*             RESOLVER_CONFIG=/etc/resolv.conf.tcp2
//*                          or
//*             _BPXK_SETIBMOPT_TRANSPORT=TCPCS
//*                          or
//*             DNS_VERSION=v4
//*
//*        Define STDENV with the name of the file that contains
//*        the environmental variables to be used for this
//*        invocation of NAMED.
//*
//*STDENV   DD PATH='/etc/named.env',
//*            PATHOPTS=(ORDONLY)
//*STDENV  DD DSN=SAMPLE.NAMED(ENV&SYSCLONE),DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
```

```
//SYSERR   DD SYSOUT=*
//SYSOUT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
```

BIND 9–DNS:

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZANSPR9
//*
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        5694-A01 (C) Copyright IBM Corp. 1997, 2000.
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//*        NAMED can be started with a variety of parameters.
//*        In this example, the "-c" parameter describes which
//*        configuration file NAMED should be started with.
//*
//NAMED    PROC C='/etc/named.conf'
//NAMED    EXEC PGM=BPXBATCH,REGION=0K,TIME=NOLIMIT,
//         PARM='PGM /usr/lpp/tcpip/sbin/named -c &C '
//*
//*        NAMED can use certain environmental variables, such
//*        as NLSPATH (to determine the location of the message
//*        catalog), and RESOLVER_CONFIG (to determine the location
//*        of the file that contains the parameter TCPIPjobname).
//*        These variables can be specified in a file defined
//*        by STDENV.
//*        An example of the contents of this file follows:
//*
//*             RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*                          or
//*             RESOLVER_CONFIG=/etc/resolv.conf.tcp2
//*                          or
//*             DNS_VERSION=v9
//*
//*        Define STDENV with the name of the file that contains
//*        the environmental variables to be used for this
//*        invocation of NAMED.
//*
//*STDENV   DD PATH='/etc/named.env',
//*            PATHOPTS=(ORDONLY)
//*STDENV  DD DSN=SAMPLE.NAMED(ENV&SYSCLONE),DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
```

Step 5. Create the domain data files (master name server only)

The domain data files contain information about a domain, such as the IP addresses and names of the hosts in the domain for which the master name server is authoritative. The *forward* domain data file contains entries that provide forward mapping (host names-to-IP addresses for each host system in the zone) as well as additional information about system resources. The *reverse* domain data file contains entries that provide reverse mapping (IP addresses-to-host names). A separate reverse domain data file for each network (or subnet) in a domain can be

created. Definition of WLM and dynamic zones is covered in “Advanced BIND 4.9.3–Name server topics” on page 482. DNS/WLM is only supported by the BIND 4.9.3 name server.

Note: The TSO user ID from which the name server is started must have the proper authority to access the name server configuration and zone files. For a complete description of file permissions within the HFS, see the *z/OS UNIX System Services Planning* (SC28–1890–02).

Naming of domain data files is flexible. For convenience in maintaining the database files, it is common to give them names such as *db.extension*, where *extension* identifies the domain of the data contained within. This document uses this convention. It also uses the .v4 and .v9 suffixes to indicate the appropriate nameserver version and the suffix *.bak* to specify a slave backup file.

Use the following to create domain data files:

- Control entries
- Resource records
- Special characters

Note: Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these files.

BIND 4.9.3–DNS: The sample forward domain file, /usr/lpp/tcpip/samples/db.mycorp.v4, is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/db.mycorp.v4.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;
;  /etc/dnsdata/db.mycorp.v4
;        name server zone data
;
$ORIGIN com.
mycorp          IN      SOA   ns1.mycorp admin.mycorp (                        {1}
             1               ; Serial (incremented when database is changed
             10800           ; Refresh (slave will check every 3 hours
             3600            ; Retry  (retry every hour after refresh failure
             604800          ; Expire (slave gives up retry after one week
```

```
            86400 )         ; Time to Live (data cached in other servers 1 day
;
$ORIGIN mycorp.com.                                                          {2}
; define domain nameservers
                   IN      NS      ns1                                       {3}
                   IN      NS      ns2

; example delagation of a subdomain
;intranet           IN      NS      ns1.intranet
;intranet           IN      NS      ns2.intranet
;ns1.intranet       IN   A    9.37.35.10
;ns2.intranet       IN   A    9.37.35.11

_http._tcp              SRV  0  0 80   www.mycorp.com.                      {4}
                        SRV  10 0 8000 www2.mycorp.com.                    {4}
_http._tcp.w3           SRV  0  0 80   www.mycorp.com.                      {5}
                        SRV  10 0 8000 www2.mycorp.com.                    {5}

localhost          IN   A    127.0.0.1
ns1                IN   A    9.37.34.10
ns2                IN   A    9.37.34.11

gateway            IN   A    9.37.34.30                                    {6}
                   IN   A    9.37.35.30                                    {6}

host1              IN   A    9.37.34.1
host2              IN   A    9.37.34.2
host3            IN   A    9.37.34.3
host4              IN   A    9.37.34.4

www2               IN   A    9.37.34.5
www                IN   A    9.37.34.6
www                IN   A    9.37.34.7

mail               IN   CNAME ns1                                           {7}
ftp                IN   CNAME ns2
```

{1}
The SOA (Start of Authority) record specifies the name server ns1 as the authoritative name server for the domain mycorp.com. The mail address of the person responsible for domain data is admin@mycorp.com.
The numbers enclosed in parentheses are parameters used to set different values for the zone.

{2}
The control entry $ORIGIN appends the string mycorp.com. to all the following host names that do not end with a dot ('.').

{3}
The NS (Name Server) records specify the name servers in the zone. Note that NS records do not distinguish between primary and secondary name servers.

{4}
The SRV records specify the location for the 'http' service using the 'tcp' protocol. The first record has a priority of 0, a weight of 0, uses port 80 and the service is provided at host, www.mycorp.com. The second record has a priority of 10 which is lower, a different port and target. A web client capable of using SRV records requesting http://mycorp.com/ would be directed to www.mycorp.com and www2.mycorp.com. The client would be responsible for determining which site to connect to first based first on priority and then on weight.

{5}
The SRV record also specifies the location for the 'http' service using the 'tcp' protocol. A web client capable of using SRV records requesting http://w3.mycorp.com/ would also be directed www.mycorp.com and www2.mycorp.com.

{6}

These A (Address) records map the host name (gateway.mycorp.com) to the IP addresses of the two networks to which it is connected.

{7}
The CNAME record specifies that the name mail is an alias for the host name ns1.mycorp.com.

BIND 9–DNS: The sample forward domain file, /usr/lpp/tcpip/samples/db.mycorp.v9, is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/db.mycorp.v9.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
;
;  /etc/dnsdata/db.mycorp.v9
;        name server zone data
;
; Default TTL value
$TTL 86400                                                                  {1}
$ORIGIN com.
mycorp         IN      SOA   ns1.mycorp admin.mycorp (                      {2}
             1               ; Serial (incremented when database is changed)
             10800           ; Refresh (slave will check every 3 hours)
             3600            ; Retry  (retry every hour after refresh failure)
             604800          ; Expire (slave gives up retry after 1 week)
             7200  )         ; Negative caching (NXDOMAIN/NXRRSET responses, 2 hrs){3}
;
$ORIGIN mycorp.com.                                                         {4}
; define domain nameservers
                   IN      NS      ns1                                      {5}
                   IN      NS      ns2

; example delagation of a subdomain
;intranet           IN      NS      ns1.intranet
;intranet           IN      NS      ns2.intranet
;ns1.intranet       IN   A    9.37.35.10
;ns2.intranet       IN   A    9.37.35.11

_http._tcp              SRV  0  0 80   www.mycorp.com.                      {6}
                        SRV  10 0 8000 www2.mycorp.com.                    {6}
_http._tcp.w3           SRV  0  0 80   www.mycorp.com.                      {7}
                        SRV  10 0 8000 www2.mycorp.com.                    {7}

localhost          IN   A    127.0.0.1
ns1                IN   A    9.37.34.10
ns2                IN   A    9.37.34.11

gateway            IN   A    9.37.34.30                                   {8}
```

```
                        IN   A    9.37.35.30

host1                   IN   A    9.37.34.1
host2                   IN   A    9.37.34.2
host3                   IN   A    9.37.34.3
host4                   IN   A    9.37.34.4

www2                    IN   A    9.37.34.5
www                     IN   A    9.37.34.6
www                     IN   A    9.37.34.7

;IPv6 addresses
www                     IN   AAAA 3ffe:8050:201:1860:42::1                    {9}
www                     IN   A6   0 3ffe:8050:201:1860:42::1                  {10}

mail                    IN   CNAME ns1                                       {11}
ftp                     IN   CNAME ns2
```

{1}
The rules for time-to-live values have been complicated somewhat in BIND v9. If named finds a $TTL directive it follows TTL semantics defined in RFC 2308, which states that records not explicitly setting a TTL inherit the TTL from the $TTL value. If there is no $TTL set, it follows TTL semantics from RFCs 1035 and 1035, which state that records with no explicit TTL inherit one from the previous record. This implies that to follow RFC 1034/1035 semantics, the SOA RR must set its TTL value. For simplicity, it is recommended that you always specify a $TTL value. This line sets the default TTL for all records to 86400 seconds (one day).

{2}
The SOA (Start of Authority) record specifies the name server ns1 as the authoritative name server for the domain mycorp.com. The mail address of the person responsible for domain data is admin@mycorp.com. The numbers enclosed in parentheses are parameters used to set different values for the zone.

{3}
The meaning of the last SOA value has changed from BIND v4 to V9. It now represents length of time other servers should cache negative responses from this zone. This line sets that value to 7200 seconds (two hours).

{4}
The control entry $ORIGIN appends the string mycorp.com. to all the following host names that do not end with a dot ('.').

{5}
The NS (Name Server) records specify the name servers in the zone. Note that NS records do not distinguish between primary and secondary name servers.

{6}
The SRV records specify the location for the 'http' service using the 'tcp' protocol. The first record has a priority of 0, a weight of 0, uses port 80 and the service is provided at host, www.mycorp.com. The second record has a priority of 10 which is lower, a different port and target. A web client capable of using SRV records requesting http://mycorp.com/ would be directed to www.mycorp.com and www2.mycorp.com. The client would be responsible for determining which site to connect to first based first on priority and then on weight.

{7}
The SRV record also specifies the location for the 'http' service using the 'tcp' protocol. A web client capable of using SRV records requesting http://w3.mycorp.com/ would also be directed www.mycorp.com and www2.mycorp.com.

{8}
These A (Address) records map the host name (gateway.mycorp.com) to the IP addresses of the two networks to which it is connected.

{9}
The AAAA IPv6 record type sets the IPv6 address of www.mycorp.com to 3ffe:8050:201:1860:42::1.

{10}
The A6 IPv6 record type is an experimental way of specifying IPv6 addresses.

```
This record sets the address of www.mycorp.com to 3ffe:8050:201:1860:42::1. The '0'
indicates that the address value is fully qualified (starts at bit 0).  See the
most recent edition of DNS and BIND by Cricket Liu and Paul Albitz (O'Reilly and
Associates, Inc.) for more information on A6 records.

{11}
    The CNAME record specifies that the name mail is an alias for the host name
ns1.mycorp.com.
```

BIND 4.9.3–DNS: The sample reverse domain file /usr/lpp/tcpip/samples/db.34.37.9.v4 is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/db.34.37.9.v4.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
;   /etc/dnsdata/db.34.37.9.v4
;
$ORIGIN 37.9.in-addr.arpa.

34 IN SOA ns1.mycorp.com. admin.mycorp.com. (
              1  10800  3600 604800 86400 )


34              IN      NS     ns1.mycorp.com.
34              IN      NS     ns2.mycorp.com.
$ORIGIN 34.37.9.in-addr.arpa.
10            IN      PTR    ns1.mycorp.com.
11            IN      PTR    ns2.mycorp.com.

1             IN      PTR    host1.mycorp.com.
2             IN      PTR    host2.mycorp.com.
3             IN      PTR    host3.mycorp.com.
4             IN      PTR    host4.mycorp.com.
5             IN      PTR    www2.mycorp.com.
6             IN      PTR    www.mycorp.com.
7             IN      PTR    www.mycorp.com.

20            IN      PTR    printserver.mycorp.com.
```

Note: Data files created locally for use by the name server are assumed to be in code page IBM-1047. For systems using other code pages, use the `iconv` command to translate from the local code page to code page IBM-1047. Refer to *z/OS UNIX System Services Command Reference* for more detailed information about this command. Files read through a network connection (for example, secondary data files) are converted to IBM-1047 by the name server before they are written to the local file system.

FTP can also be used to convert the files to code page IBM-1047.

BIND 9–DNS: The sample reverse domain file /usr/lpp/tcpip/samples/db.34.37.9.v9 is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/db.34.37.9.v9.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
;   /etc/dnsdata/db.34.37.9.v9
;
; Default TTL value
$TTL 86400
$ORIGIN 37.9.in-addr.arpa.

34 IN SOA ns1.mycorp.com. admin.mycorp.com. (
```

```
                              1  10800  3600 604800 7200 )


34              IN      NS     ns1.mycorp.com.
34              IN      NS     ns2.mycorp.com.
$ORIGIN 34.37.9.in-addr.arpa.
10            IN      PTR    ns1.mycorp.com.
11            IN      PTR    ns2.mycorp.com.

; Build similar reverse lookup records
$GENERATE 1-4 $ PTR host$.mycorp.com.                                    {1}
; The following records are generated by the above $GENERATE directive.
;1             IN      PTR    host1.mycorp.com.
;2             IN      PTR    host2.mycorp.com.
;3             IN      PTR    host3.mycorp.com.
;4             IN      PTR    host4.mycorp.com.
5             IN      PTR    www2.mycorp.com.
6             IN      PTR    www.mycorp.com.
7             IN      PTR    www.mycorp.com.

20            IN      PTR    printserver.mycorp.com.

{1}
 $GENERATE is a v9-specific directive that is useful for creating a series of
records that differ only by an iterator.  This line prompts the name server to
create the records listed below upon zone load.  For more information on
$GENERATE, refer to the z/OS Communications Server:  IP Configuration Reference.
```

Note: Data files created locally for use by the name server are assumed to be in code page IBM-1047. For systems using other code pages, use the `iconv` command to translate from the local code page to code page IBM-1047. Refer to *z/OS UNIX System Services Command Reference* for more detailed information about this command. Files read through a network connection (for example, secondary data files) are converted to IBM-1047 by the name server before they are written to the local file system.

FTP can also be used to convert the files to code page IBM-1047.

Step 6. Create the hints (root server) file

The hints file contains the names and IP addresses of the authoritative root domain name servers. The root name servers contain the names of name servers in the top-level domains such as com, edu, and mil. The name server uses root server information when deciding which name server to contact when it receives a query for a host outside its zone of authority and it does not have the data in its cache.

Note: The hints file does not contain cached data nor does the name server provide other hosts with the information contained in the hints file. A forward-only server is the only type of name server that does not require a hints file.

To obtain a hints file, point your Web browser at `ftp://ftp.rs.internic.net` and retrieve the file `named.root` from the `domain` subdirectory. Update your hints file on a regular basis.

The `cache` directive in a BIND 4.9.3 boot file specifies the path and name of the hints file.

The hints file in a BIND 9 config file is specified with a zone{} statement of type 'hints'.

An example of a hints file originally copied from ftp://ftp.rs.internic.net/domain/named.root is listed below. Continuing the example that began in the boot and conf files, the file would be /etc/dnsdata/db.cache.

```
;   This file holds the information on root name servers needed to
;   initialize cache of Internet domain name servers
;   (e.g., reference this file in the "cache   .   <file>"
;   or zone "." { type hint; file "db.cache"; };
;   in a v4 or v9 config file, respectively.
;   configuration file of BIND domain name servers).
;
;   This file is made available by InterNIC registration services
;   under anonymous FTP as
;   file                /domain/named.root
;   on server           FTP.RS.INTERNIC.NET
;   -OR- under Gopher** at RS.INTERNIC.NET
;       under menu      InterNIC Registration Services (NSI)
;         submenu       InterNIC Registration Archives
;       file            named.root
;
;   last update:       May 19, 1997
;   related version of root zone:      1997051700
;
;   formerly NS.INTERNIC.NET
;
.                     3600000   IN  NS   A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.   3600000       A    198.41.0.4
;
;   formerly NS1.ISI.EDU
;
.                     3600000       NS   B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.   3600000       A    128.9.0.107
;
;   formerly C.PSI.NET
;
.                     3600000       NS   C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.   3600000       A    192.33.4.12
;
;   formerly TERP.UMD.EDU
;
.                     3600000       NS   D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.   3600000       A    128.8.10.90
;
;   formerly NS.NASA.GOV
;
.                     3600000       NS   E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.   3600000       A    192.203.230.10
;
;  formerly. NS.ISC.ORG
;
.                     3600000       NS   F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.   3600000       A    192.5.5.241
;
;   formerly NS.NIC.DDN.MIL
;
.                     3600000       NS   G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.   3600000       A    192.112.36.4
;
;   formerly AOS.ARL.ARMY.MIL
;
.                     3600000       NS   H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.   3600000       A    128.63.2.53
;
;   formerly NIC.NORDU.NET
;
.                     3600000       NS   I.ROOT-SERVERS.NET.
```

```
I.ROOT-SERVERS.NET.   3600000       A    192.36.148.17
;
;   temporarily housed at NSI (InterNIC)
;
.                     3600000       NS   J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.   3600000       A    198.41.0.10
;
;   housed in LINX, operated by RIPE NCC
;
.                     3600000       NS   K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.   3600000       A    193.0.14.129
;
;   temporarily housed at ISI (IANA)
;
.                     3600000       NS   L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.   3600000       A    198.32.64.12
;
;   temporarily housed at ISI (IANA)
;
.                     3600000       NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.   3600000       A    198.32.65.12
; End of File
```

Step 7. Create the loopback file

The loopback file contains the loopback address. This is the address that a host uses to route queries to itself. The preferred loopback address is `127.0.0.1`, although you can configure additional loopback interfaces in the PROFILE.TCPIP. For BIND 4.9.3, DNS will bind to `127.0.0.1` in addition to the first loopback address configured in PROFILE.TCPIP. BIND 9 mode requires the availability to bind onto loopback address 127.0.0.1.

BIND 9 and BIND 4 server modes coexistence on the same host requires different loopback addresses. If BIND 9 and BIND 4.9.3 are to be started on the same host, once BIND 9 mode is started, BIND 4 mode server will have to use a different loopback address. BIND 4 server master forward and reverse zone files should define A and PTR resource records, respectively, for the appropriate loopback addresses.

This guide uses the extension `.loopback` to specify the loopback file.

Note: In addition to creating the loopback file, add an address resource record called *localhost* to the forward domain data file. This record supports proper two-way resolution.

Use the following elements to create the loopback file:

- Control entries
- Resource records
- Special characters

Note: Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these files.

BIND 4.9.3–DNS: The sample loopback file for BIND 4.9.3 shipped in /usr/lpp/tcpip/samples/db.loopback.v4 is listed below. Continuing the example that started in the boot and .conf files, the file would be /etc/dnsdata/db.loopback.v4.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
;   /etc/dnsdata/db.loopback.v4
```

```
;
0.0.127.in-addr.arpa. IN SOA  ns1.mycorp.com. admin.mycorp.com. (
     1
     10800
     3600
     604800
     86400   )

0.0.127.in-addr.arpa.   IN   NS  ns1.mycorp.com.
0.0.127.in-addr.arpa.   IN   NS  ns2.mycorp.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
```

For DNS/WLM and BIND 4.9.3: To get the sysplex domain name, add the following PTR record to the loopback file in the loopback zone:

128.0.0.127.in-addr.arpa. IN PTR Sysplex_Domain_Name.

Sysplex_Domain_Name is the domain name of the sysplex (specified as cluster zone in the master boot file). Do not forget to put a period (.) after the Sysplex_Domain_Name. Note that all of the sysplex name servers must be updated with this change.

BIND 9–DNS: The sample loopback file for BIND 9 shipped in /usr/lpp/tcpip/samples/db.loopback.v9 is listed below. Continuing the example that started in the boot file, the file would be /etc/dnsdata/db.loopback.v9.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
;   /etc/dnsdata/db.loopback.v9
;
; Default TTL value
$TTL 86400
0.0.127.in-addr.arpa. IN SOA  ns1.mycorp.com. admin.mycorp.com. (
     1
     10800
     3600
     604800
     7200    )

0.0.127.in-addr.arpa.   IN   NS  ns1.mycorp.com.
0.0.127.in-addr.arpa.   IN   NS  ns2.mycorp.com.
1.0.0.127.in-addr.arpa. IN   PTR localhost.
```

A separate loopback file is required for use with the IPv6 loopback address (::1). The following shows an example named.conf configuration and the associated zone file.

```
IPv6 loopback master zone simple definition in named.conf: zone
"1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" { type master; file "loopback.v6"; };

loopback.v6 zone file (with implied domain origin from above master zone definition)

$TTL 86400
@   IN SOA ns1.mycorp.com. admin.mycorp.com. (
     1
     10800
     3600
     604800
     7200 )
        NS ns1.mycorp.com.
        PTR localhost.
```

Step 8. For BIND 9 only — configure logging

A wide variety of logging options for the nameserver can be configured via the *logging* statement. Its *channel* phrase associates output methods, format options

and severity levels with a name that can then be used with the *category* phrase to select how various classes of messages are logged.

Only one *logging* statement is used to define as many channels and categories as are wanted. If there is no logging statement, the logging configuration will be:

```
logging {
     category "default" { "default_syslog"; "default_debug"; };
  };
```

In BIND 9, the logging configuration is only established when the entire configuration file has been parsed. When the server is starting up, all logging messages regarding syntax errors in the configuration file go to the default channels. Therefore, if started from a procedure, the logging messages will be written to syslogd. If started from z/OS UNIX, the logging messages may be written to 'named.run' if started with the –d option, in addition to syslog.

All log output goes to one or more channels; you can make as many of them as you want. Every channel definition must include a clause that says whether messages selected for the channel go to a file, to a particular syslog facility, or are discarded. It can optionally also limit the message severity level that will be accepted by the channel (the default is *info*), and whether to include a named-generated time stamp, the category name, the severity level, and the thread ID (the default is to include all).

Messages written to logging files can be buffered according to the value on the *max-buffered-messages* options statement. The default value is the maximum allowed value of 35. Buffering messages to logging files will provide some amount of a performance advantage. However, it might be misleading when viewing a logging file while the name server is running, since the most recent logging information might not have been written yet to the file.

The word *null* specified as the destination option for the channel will cause all messages sent to it to be discarded; in that case, other options for the channel are meaningless.

The *file* names the HFS path name for the log file and can include limitations, both on how large the file is allowed to become and how many versions of the file will be saved each time the file is opened.

The *size* option for files is simply a hard ceiling on log growth. If the file ever exceeds the size and the version is zero, then named will not write anything more to it until the file is reopened, which will be done after the file is renamed or erased. If version option value is 1 or more, the current file, when full, is renamed with a suffix and another file is opened with the original name. In the latter case, logging will continue in a round robin fashion using the current file name and the suffixed file names. The default behavior is not to limit the size of the file. Note that if debug is enabled, the logs can grow large very quickly and you may run the risk of filling up your HFS. Therefore, it is recommended that you limit the size of the file when debugging is enabled.

If you use the *version* log file option, then named will retain that many backup versions of the file by renaming them when opening. For example, if you choose to keep 3 old versions of the file *lamers.log* then just before it is opened *lamers.log.1* is renamed to *lamers.log.2* , *lamers.log.0* is renamed to *lamers.log.1* , and

lamers.log is renamed to *lamers.log.0* . No rolled versions are kept by default; any existing log file is simply appended. The *unlimited* keyword is synonymous with 99 in current BIND releases.

Example usage of the size and versions options:

```
channel "an_example_channel" {
        file "example.log" versions 3 size 20m;
        print-time yes;
        print-category yes;
        print-threadid yes;
    };
```

The argument for the *syslog* clause is a syslog facility. Refer to the *z/OS Communications Server: IP Configuration Reference* for more detailed parameter information.

The severity clause works like syslog's priorities, except that they can also be used if you are writing straight to a file rather than using syslog . Messages which are not at least of the severity level given will not be selected for the channel; messages of higher severity levels will be accepted. Severity level decreases from critical down to info, and further decreases from debug 1 down to debug 99.

If you are using syslog , then the syslog.conf priorities will also determine what eventually passes through. For example, defining a channel facility and severity as *daemon* and *debug* but only logging daemon.warning via syslog.conf will cause messages of severity *info* and *notice* to be dropped. If the situation were reversed, with named writing messages of only warning or higher, then syslogd would print all messages it received from the channel.

The server can supply extensive debugging information when it is in debugging mode. If the server's global debug level is greater than zero, then debugging mode will be active. The global debug level is set by starting the named server with the -d flag followed by a positive integer. All debugging messages in the server have a debug level, and higher debug levels give more detailed output. The maximum debug level is 99. Channels that specify a specific debug severity, for example:

```
channel "specific_debug_level" {
      file "foo";
      severity debug 3;
  };
```

will get debugging output of level 3 or less any time the server is in debugging mode, regardless of the global debugging level. Channels with *dynamic* severity use the server's global level to determine what messages to print.

The *print–* options can be used in any combination.

If the following is turned on . . .	Then the following is logged . . .
print–time	date and time
print–category	category of the message
print–severity	severity level of the message
print–threadid	the thread ID that is issuing the message

Note: *print–time* can be specified for a syslog channel, but is usually pointless since syslog also prints the date and time.

The *print–* options are always printed in the following order: time, category, severity, and thread ID. Here is an example:

```
Apr 24 09:28:05.848 queries: info: 0a923850: EZZ8828I client 127.0.0.1#20021:
                    query: host1.mycorp.com IN A
```

There are four predefined channels that are used for named ’s default logging as follows:

```
channel "default_syslog" {
        syslog daemon;                          // end to syslog's daemon
                                                // facility
        severity info;                          // only send priority info
                                                // and higher
    };
    channel "default_debug" {
        file "named.run";                       // write to named.run in
                                                // the working directory
                                                // Note: stderr is used instead
                                                // of "named.run"
                                                // if the server is started
                                                // with the '-f' option.
        severity dynamic                        // log at the server's
                                                // current debug level
    };
    channel "default_stderr" {                  // writes to stderr
        file "<stderr>";                        // this is illustrative only;
                                                // there's currently no way of
                                                // specifying an internal file
                                                // descriptor in the
                                                // configuration language.
        severity info;                          // only send priority info
                                                // and higher
    };
    channel "null" {
       null;                                    // toss anything sent to
                                                // this channel
 };
```

The *default_debug* channel normally writes to a file named.run in the server’s working directory. For security reasons, when the -u command line option is used, the named.run file is created only after named has changed to the new UID, and any debug output generated while named is starting up and still running as root is discarded.

Once a channel is defined, it cannot be redefined. Thus you cannot alter the built-in channels directly, but you can modify the default logging by pointing categories at channels you have defined.

There are many categories, so you can send the logs you want to see wherever you want, without seeing logs you do not want. If you don’t specify a list of channels for a category, then log messages in that category will be sent to the *default* category instead. If you don’t specify a default category, the following ″default default″ is used:

```
category "default" { "default_syslog"; "default_debug"; };
```

As an example, say you want to log security events to a file, but you also want keep the default logging behavior. You would specify the following:

```
channel "my_security_channel" {
    file "my_security_file";
    severity info;
};
category "security" {
```

```
    "my_security_channel";
    "default_syslog";
    "default_debug";
};
```

To discard all messages in a category, specify the null channel:

```
category "xfer-out" { "null"; };
category "notify" { "null"; };
```

In the following example:

- Four channels have been defined to make it possible to browse and keep logs for some categories separately. In theory, each category can log to one or more different channels, but keeping the number of channels to a minimum is recommended.
- Most existing categories have been specifically associated with one or more channels to demonstrate logging flexibility. However, categories directed to the main log only may be omitted and instead, covered by the default category. One exception is the *queries* category, which requires a specific channel association to enable queries logging.
- Every channel log entry will be prefixed with time stamp, category, severity, and thread ID.

  Note: The default for all print- options is *yes*.
- Every channel has been customized for maximum file size and for keeping 2 archived files in addition to the active log file. Make sure the disk has enough space for the total maximum size of active and archived log files, and extra space for any other growing logs or files.
- Every channel but transfer_log will log messages up to debug level 99 which is the suggested detailed level to gather problem documentation but can fill up logging files quickly. Lower debug levels (e.g. 11, info, error) may be used for normal operation.
- - transfer_log is shown at debug level 7, where minimal zone transfer starting and stopping activity is recorded. Increasing level to 8 and above will considerably increase logging activity, mainly for large zones with one-answer transfer format.
- ″severity dynamic;″ can also be set for any channel, in which case debug level is determined by named -d start option value.
- The default_debug channel can be specified instead of one or more user defined channels, in which case logging goes to ″named.run″ file in the named working directory. No maximum file size will stop logging to named.run.

```
logging {
 channel main_log {
      file "/tmp/named_main.log" versions 2 size 20M;
      print-time yes;
      print-category yes;
      print-severity yes;
      print-threadid yes;
      severity debug 99;
                  };
 channel security_log {
      file "/tmp/named_security.log" versions 2 size 1M;
      severity info;
      severity debug 99;
                    };
 channel query_log {
      file "/tmp/named_query.log" versions 2 size 10M;
      severity info;
```

```
      severity debug 99;
                  };
 channel transfer_log {
      file "/tmp/named_transfer.log" versions 2 size 10M;
      severity debug 7;
                       };


     category client   { main_log; };
     category config   { main_log; };
     category database { main_log; };
     category dispatch { main_log; };
     category dnssec { security_log; main_log; };
     category general  { main_log; };
     category network  { main_log; };
     category notify   { main_log; };
     category resolver { main_log; };
     category security { security_log; main_log; };
     category update   { main_log; };
     category queries  { query_log;};
     category lame-servers { query_log; main_log; };
     category xfer-in  { "transfer_log"; };
     category xfer-out { "transfer_log"; };
     category default  { main_log; };
     category unmatched { main_log; };
        };
```

More detail about the available categories and brief descriptions of the types of log information they contain can be found in the *z/OS Communications Server: IP Configuration Reference*.

Step 9. Ensure that the syslog daemon is running on your system

The name server uses the syslog daemon to log messages. To verify that the name server starts correctly or to diagnose problems, the syslog daemon should be running.

If your syslog daemon is not configured, see "Creating the syslog file" on page 458 for information regarding the syslog daemon.

Step 10. Specify whether the name server is to run swappable or nonswappable

You might want to run the name server in a swappable state, as it has in the past. This is an optional step. Keep in mind that when an application makes an address space nonswappable, it might convert additional real storage in the system to preferred storage. Because preferred storage cannot be configured offline, allowing the name server to run in a nonswappable state can reduce the installation's ability to reconfigure storage in the future.

If you want to run the name server as swappable, you must have the ″BPX.STOR.SWAP″ FACILITY class profile defined to RACF with no universal access. To do this, enter the following commands from a RACF user ID.

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
SETROPTS RACLIST(FACILITY) REFRESH
```

The name server will also run as swappable if the ″BPX.STOR.SWAP″ FACILITY is not defined and the name server is started from a user ID with a UID not equal to 0.

If you want the name server to run in a nonswappable state, do one of the following:

- Do not define the BPX.STOR.SWAP facility to RACF and start the name server from a user ID with a UID=0.
- Define the facility to RACF and allow the appropriate users at least READ access to the facility.

The latter method can be accomplished with the following set of commands:

```
RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)
PERMIT BPX.STOR.SWAP CLASS(FACILITY) ID(userid) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Step 11. Start the name server

Your name server is ready to start. Start the name server using the following methods:

- A supervisor with an authorized TSO ID can start a name server from the MVS operator's console by starting the named start procedure. If the boot or config file path is not `/etc`, specify the correct path in the start procedure. Refer to *z/OS Communications Server: IP Configuration Reference* for start options. A sample start procedure is provided with the product and is found in SEZAINST(NAMED4) for BIND 4.9.3 and in SEZAINST(NAMED9) for BIND 9.
- A user ID with superuser authority can start the name server from the shell by starting z/OS UNIX, then issuing the `named` command and, optionally, any parameters.
- It is also possible to start the server automatically when z/OS UNIX is started by specifying the path and file name of the z/OS UNIX initialization shell script in the `/etc/init.options` file using the `-sc` option:

  ```
  -sc /etc/rc       shell script = /etc/rc
  ```

  The file `/etc/rc` is the default z/OS UNIX initialization shell script that is executed when z/OS UNIX is started. See *z/OS Communications Server: IP Migration* for more detail. Information such as the following can be entered in `/etc/rc`:

  ```
  FOR BIND 4.9.3-DNS

  # Start name server
  /usr/lpp/tcpip/sbin/named -b /named/production/named.boot &

  FOR  BIND 9-DNS

  # Start name server
  /usr/lpp/tcpip/sbin/named -c /named/production/named.conf &
  ```

  Port 53 may be reserved for the name server in the PROFILE.TCPIP data set. For directions on specifying port ownership, see "Step 3. Specify port ownership" on page 431. Only when in BIND 4 mode, when the stack to which named binds is started, named completes initialization. The BIND 9 name server may only be started after TCP/IP is up. In rare circumstances, the BIND 9 name server may complete initialization before all of the stack's interfaces have been brought up. In this case, the name server will not be listening on all desired interfaces. Eventually, the name server will scan the stack interfaces again, and begin listening on all desired interfaces. The default time period for this rescan is one minute. You can have the name server rescan the interfaces at the interval you desire by specifying the ″interface-interval″ option in the named.conf file.

- If you are starting the BIND 4.9.3 name server, use the AUTOLOG statement to start the name server automatically during initialization with z/OS UNIX running. Insert the name of the named start procedure in the AUTOLOG statement of the PROFILE.TCPIP data set.

  ```
  AUTOLOG
    NAMED
  ENDAUTOLOG
  ```

  The JOBNAME keyword should not be added to the AUTOLOG statement for named.

  If you are starting the BIND 9 name server, use some other automation outside of AUTOLOG to automatically start the name server since it is a generic server. For more information on the AUTOLOG statement, refer to *z/OS Communications Server: IP Configuration Reference*.

  Note: Named cannot be started from `INETD`.

Step 12. Verify that the name server started correctly

BIND 4.9.3–DNS: After starting the name server, ensure that no errors occurred when it was started. Look at the syslog daemon output data set for name server messages. If startup is successful, messages similar to the following are displayed:

```
named[22]: EZZ6698I name server starting. @(#) ddns/ns/ns_main.c,
           dns_ns, dns_r1.1 1.62 9/23/97 10:57:21
named[22]: EZZ6701I named established affinity with 'TCPCS'
named[22]: EZZ6540I Static primary zone 'raleigh.ibm.com' loaded (serial 1)
named[22]: EZZ6540I Static primary zone '34.37.9.inaddr.arpa' loaded (serial 1)
named[22]: EZZ6540I Static primary zone '0.0.127.inaddr.arpa' loaded (serial 1)
named[22]: EZZ6540I Static cache zone '' loaded (serial 0)
named[23]: EZZ6475I named: ready to answer queries.
```

To correct errors, either stop and restart the name server to pick up the changes, or reload the name server with the -SIGHUP signal.

To stop the name server from the z/OS UNIX shell, issue:

```
kill -TERM $(cat /etc/named.pid)
```

To stop the name server from the MVS console, issue the following:

```
p named3
(Use the name of the procedure that is currently active. This is
usually the proc name that was used to start the name server, followed
by a '1'  for  BIND 9 server, by a '3' for BIND 4 server, due to extra forking
steps on startup)
```

To reload the BIND 4.9.3 name server with a signal, issue the following command from the z/OS UNIX shell:

```
kill -HUP $(cat bind9_pid_file)
```

where *named_pid_path* is derived from pid-file option in named v9 configuration file.

BIND 9–DNS: After starting the name server, ensure that no errors occurred when it was started. Look at the syslog daemon output data set for name server messages. If startup is successful, messages similar to the following are displayed:

```
Mar 26 ... mvsw named[...29]: EZZ9172I VM mode detected. Using 1 CPU(s) for -n option
Mar 26 ... mvsw named[...56]: EZZ9547I starting named, BIND 9.2.0 -c /etc/namedgm.conf
Mar 26 ... mvsw named[...56]: EZZ9095I STARTING NAMED, BIND 9.2.0
Mar 26 ... mvsw named[...56]: EZZ9217I Running non-swappable
Mar 26 ... mvsw named[...56]: EZZ9540I using 1 CPU
```

```
Mar 26 ... mvsw named[...56]: EZZ9126I loading configuration from '/etc/namedgm.conf'
Mar 26 ... mvsw named[...56]: EZZ8842I the default for the 'auth-nxdomain' option is now 'no'
Mar 26 ... mvsw named[...56]: EZZ9052I no IPv6 interfaces found
Mar 26 ... mvsw named[...56]: EZZ9046I listening on IPv4 interface VLINK1, 9.67.116.122#53
Mar 26 ... mvsw named[...56]: EZZ9046I listening on IPv4 interface TR1, 9.67.113.75#53
Mar 26 ... mvsw named[...56]: EZZ9046I listening on IPv4 interface loopback127, 127.0.0.1#53
Mar 26 ... mvsw named[...56]: EZZ9111I command channel listening on 9.67.113.75#953
Mar 26 ... mvsw named[...56]: EZZ9130I NAMED, BIND 9.2.0 IS RUNNING
```

To stop the name server from the z/OS UNIX shell, issue:

```
kill -TERM $(cat /etc/named.pid)
```

To stop the name server from the MVS console, issue the following:

```
p named1
(Use the name of the procedure that is currently active. This is
usually the proc name that was used to start the name server, followed
by a '1'  for  BIND 9 server, by a '3' for BIND 4 server, due to extra
forking steps on startup)
```

To reload the BIND 9 name server with a signal, issue the following command from the z/OS UNIX shell:

```
kill -HUP $(cat bind9_pid_file)
```

rndc can also be used to reload or stop the server. Refer to “Remote Name Daemon Control (rndc)” on page 458 for more details on the rndc command.

Step 13. Verify the name server can accept queries

When the name server is up with no logged errors, ensure that it can accept queries. Ensure that the name server can accept queries locally from both the MVS and z/OS UNIX environments. In order to correctly set up these environments, see “Understanding search orders of configuration information” on page 18 for instructions.

After the resolver configuration is correct, test with the nslookup or dig command. An example using nslookup follows.

Issue the following command from both the z/OS UNIX shell and the TSO ready prompt. In the following example, the name ’host1.mycorp.com.’ is used for the search.

Note: Choose any name in the domain you have defined.

```
nslookup host1.mycorp.com
```

Using the sample files in this example, the following should be the result when the command is issued:

```
 $ nslookup host1.mycorp.com
 Defaulting to nslookup version 4
 Starting nslookup version 4
 Server:  localhost
 Address:  127.0.0.1

 Name:    host1.mycorp.com
 Address:  9.37.34.1
```

Configuring a slave name server

After setting up a working master name server, one or more slave name servers can be set up. This process is very similar as configuring a master name server. The differences are in the boot file for BIND 4.9.3 and the conf file for BIND 9 and the absence of the domain data files.

For example, see “Configuring a master (primary) name server” on page 427 to configure a slave server for the forward and reverse mapping zones. The steps are identical to the steps for configuring a master name server, except for step 1. For step 1, more information is included below in the subsections following the steps.

1. Create a configuration file for your environment:
   a. Create the boot file for BIND 4.9.3–DNS. See “Step 1a. Create the boot file for BIND 4.9.3–DNS”.
   b. Create the configuration file for BIND 9–DNS. See “Step 1b. Create the configuration file for BIND 9–DNS.” on page 452.
2. See “Step 2. For BIND 4.9.3–DNS only: specify stack affinity (Multiple stack environment)” on page 431.
3. See “Step 3. Specify port ownership” on page 431.
4. See “Step 4. Update the name server start procedure (Optional)” on page 432.
5. See “Step 5. Create the domain data files (master name server only)” on page 433.
6. See “Step 6. Create the hints (root server) file” on page 439.
7. See “Step 7. Create the loopback file” on page 441.
8. See “Step 8. For BIND 9 only — configure logging” on page 442.
9. See “Step 9. Ensure that the syslog daemon is running on your system” on page 447.
10. See “Step 10. Specify whether the name server is to run swappable or nonswappable” on page 447.
11. See “Step 11. Start the name server” on page 448.
12. See “Step 12. Verify that the name server started correctly” on page 449.
13. See “Step 13. Verify the name server can accept queries” on page 450.

The difference between configuring a master and slave name server is the creation of domain data files (the database files containing host-to-address and address-to-host mappings). The domain data files are maintained on the master name server, and the slave name server transfers this data to its own database.

Instructions for creating the boot file for a slave name server are below. All remaining steps are identical to those in “Configuring a master (primary) name server” on page 427.

Step 1a. Create the boot file for BIND 4.9.3–DNS

The easiest way to create a boot file for a slave name server is to start from the boot file for the master name server. The sample boot file, /usr/lpp/tcpip/samples/slave.boot (based on named.boot), reflects the setup for a slave name server and is shown below.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2001
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
```

```
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
; Note:  This file must be copied and renamed to /etc/named.boot and
;        all zone files referenced below must be copied to /etc/dnsdata/ for
;        this file to function as intended.  Also note this example is
;        built on the assumption that there is a master server for the zones
;        at 9.37.34.10.
;
;  /etc/named.boot
;
;        boot file for name server
;
;
;type    domain                  source file or host
;
directory   /etc/dnsdata
secondary   mycorp.com           9.37.34.10     db.mycorp.bak
secondary   34.37.9.in-addr.arpa 9.37.34.10     db.34.37.9.bak
primary     0.0.127.in-addr.arpa db.loopback.v4
cache       .                    db.cache
options query-log
```

This boot file specifies:

1. The location of the files (/etc/dnsdata).
2. The name server will be the slave name server for the `mycorp.com` zone.
3. The primary name server is located at IP address `9.37.34.10`.
4. The data for `mycorp.com` will be stored in /etc/dnsdata/db.mycorp.bak after it is retrieved from the primary name server by doing a zone transfer.
5. The name server will be the slave name server for the reverse mapping zone, `34.37.9.in-addr.arpa`.
6. The primary name server is located at IP address `9.37.34.10`.
7. The data for addresses `9.37.34.x` contained in the zone will be stored in /etc/dnsdata/db.34.37.9.bak after it is retrieved from the primary name server by doing a zone transfer.
8. The name server will continue to function in the primary role for the loopback address (`127.0.0.1`).
9. The list of root name servers is in /etc/dnsdata/db.cache.
10. All queries coming in to this name server will be logged in the syslog daemon output file.

Step 1b. Create the configuration file for BIND 9–DNS.

This example illustrates the equivalent configuration for a v9 name server where the sample configuration file, /usr/lpp/tcpip/samples/slave.conf (based on named.conf), reflects the setup for a slave name server.

```
#           LICENSED MATERIALS - PROPERTY OF IBM
#             "RESTRICTED MATERIALS OF IBM"
#          5694-A01 (C) COPYRIGHT IBM CORP. 2000
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 1993
# All Rights Reserved
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
```

```
# Licensed Materials - Property of IBM
#
#
#       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
#
# INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
# EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
# WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
# LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
# OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
# IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
# YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
# CORRECTION.
#
# Note:  This file must be copied and renamed to /etc/named.conf and all
#        all zone files referenced below must be copied to /etc/dnsdata/ for
#        this file to function as intended.  Also note this example is
#        built on the assumption that there is a master server for the zones
#        at 9.37.34.10.  In addition, the default location for the process
#        id file is in /var/run/pid.file; if that directory does not exist
#        a different one can be configured with the option:
#
#        pid-file "path/file-name";
#
#  /etc/named.conf
#
#        conf file for name server
#
options {
 directory "/etc/dnsdata";
};

logging {
 category "queries" {
  default_syslog;
 };
};

zone "mycorp.com" in {
 type slave;
 file "db.mycorp.bak";
 masters { 9.37.34.10; };
};

zone "34.37.9.in-addr.arpa" in {
 type slave;
 file "db.34.37.9.bak";
 masters { 9.37.34.10; };
};

zone "0.0.127.in-addr.arpa" in {
 type master;
 file "db.loopback.v9";
};

zone "." in {
 type hint;
 file "db.cache";
};
```

Configuring a cache-only name server

If the name server does not need to be authoritative for any data, choose a special type of name server, called a caching-only name server. A caching-only name

server can improve performance by reducing the number of network flows required for names or addresses that are frequently requested.

Use the following steps to configure a basic name server. More information for step 1 is included in the subsections following the steps. For the subsequent steps, see the appropriate sections elsewhere in the book.

1. Create a configuration file for your environment:
   a. Create the boot file for BIND 4.9.3–DNS. See “Step 1a. Create the boot file for BIND 4.9.3–based DNS.”.
   b. Create the configuration file for BIND 9–DNS. See “Step 1b. Create the configuration file for BIND 9–DNS.” on page 455.
2. See “Step 2. For BIND 4.9.3–DNS only: specify stack affinity (Multiple stack environment)” on page 431.
3. See “Step 3. Specify port ownership” on page 431.
4. See “Step 4. Update the name server start procedure (Optional)” on page 432.
5. See “Step 6. Create the hints (root server) file” on page 439.

   Following is an example of a hints (root server) file for a cache-only server:

   ```
   ;
   ;  Cache-only "hints" file
   ;
   .                       3600000 IN NS   hostname
   hostname.               3600000    A    ipaddress
   ```

   where *hostname* is the fully qualified host name of an authoritative name server for the root (’.’) domain and *ipaddress* is the IP address for the specified hostname. How this file is configured depends on whether you are behind a firewall or not. If behind a firewall, hostname should be the name of an internal root name server if internal roots are being used. If you are behind a firewall and not using internal roots, then requests are probably being forwarded to a name server on a bastion host, which can resolve internal and internet names. In the latter case, what is in the hints file is unimportant since it will not be used, and if the name server does attempt to use it, the firewall would block it from contacting the internet root name servers. If you are not behind a firewall, follow the example in “Step 6. Create the hints (root server) file” on page 439 and instructions on getting a recent copy of the internet root name servers.
6. See “Step 7. Create the loopback file” on page 441.
7. See “Step 8. For BIND 9 only — configure logging” on page 442.
8. See “Step 9. Ensure that the syslog daemon is running on your system” on page 447.
9. See “Step 10. Specify whether the name server is to run swappable or nonswappable” on page 447.
10. See “Step 11. Start the name server” on page 448.
11. See “Step 12. Verify that the name server started correctly” on page 449.
12. See “Step 13. Verify the name server can accept queries” on page 450.

The difference between configuring a master name server and configuring a caching-only server is the creation of domain data files (the database files containing host-to-address and address-to-host mappings). A caching-only name server will only contain the loopback zone file and the hints file.

Step 1a. Create the boot file for BIND 4.9.3–based DNS.

The easiest way to create a boot file for a caching-only name server is to use the boot file for the master name server. The sample boot file

/usr/lpp/tcpip/samples/caching.boot (based on named.boot), reflects the setup for a caching-only name server and is shown below.

```
;           LICENSED MATERIALS - PROPERTY OF IBM
;             "RESTRICTED MATERIALS OF IBM"
;          5694-A01 (C) COPYRIGHT IBM CORP. 2000
;
; (C) COPYRIGHT International Business Machines Corp. 1985, 1993
; All Rights Reserved
; US Government Users Restricted Rights - Use, duplication or
; disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
;
; Licensed Materials - Property of IBM
;
;
;       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
;
; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
; WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
; LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
; OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
; IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
; YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
; CORRECTION.
;
; Note:  This file must be copied and renamed to /etc/named.boot and all
;        all zone files referenced below must be copied to /etc/dnsdata/ for
;        this file to function as intended.
;
;  /etc/named.boot
;
;        boot file for name server
;
;
;type    domain                  source file or host
;
directory /etc/dnsdata                              {1)
primary  0.0.127.in-addr.arpa  db.loopback.v4       {2}
cache  .                       db.cache             {3}
options query-log                                   {4}
```

This boot file specifies:

1. The location of the files (/etc/dnsdata).
2. The name server will continue to function in the primary role for the loopback address (127.0.0.1).
3. The list of root name servers is in /etc/dnsdata/db.cache.
4. All queries coming in to this name server will be logged in the syslog daemon output file.

Step 1b. Create the configuration file for BIND 9–DNS.

The following sample configuration sets up an equivalent configuration for a BIND 9 nameserver where the sample configuration file, /usr/lpp/tcpip/samples/caching.conf (based on named.conf), reflects the setup for a caching-only name server.

```
#           LICENSED MATERIALS - PROPERTY OF IBM
#             "RESTRICTED MATERIALS OF IBM"
#          5694-A01 (C) COPYRIGHT IBM CORP. 2000
#
#(C) COPYRIGHT International Business Machines Corp. 1985, 1993
# All Rights Reserved
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
```

```
# Licensed Materials - Property of IBM
#
#
#       NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
#
# INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
# EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT
# WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT
# LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
# OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
# IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE
# YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR
# CORRECTION.
#
#
# Note:  This file must be copied and renamed to /etc/named.conf and all
#        all zone files referenced below must be copied to /etc/dnsdata/ for
#        this file to function as intended.  In addition, the default location
#        for the process id file is in /var/run/pid.file; if that directory
#        does not exist a different one can be configured with the option:
#
#        pid-file "path/file-name";
#
#  /etc/named.conf
#
#        conf file for name server
#
options {
 directory "/etc/dnsdata";
};

logging {
 category "queries" {
  default_syslog;
 };
};

zone "0.0.127.in-addr.arpa" in {
 type master;
 file "db.loopback.v9";
};

zone "." in {
 type hint;
 file "db.cache";
};
```

Configuring a stealth name server

A stealth server is a server that answers authoritatively for a zone, but is not listed in that zone's NS records. Configure a master or slave stealth server for a zone like you would configure a visible master or slave server for the zone, except do not create an NS record for the stealth server in the master zone. The named server configuration option *also-notify* may be used to notify stealth slave servers of a zone update. In order to configure a stealth server, follow the steps in "Configuring a slave name server" on page 450.

Adding forwarding to your name server

In order to use forwarding in any BIND 4.9.3 or BIND 9 name server, update the boot file for BIND 4.9.3 or the conf file for BIND 9.

Add the following statement to the BIND 4.9.3 boot file:

```
forwarders 9.4.2.1
```

Add the following statement to the BIND 9 conf file in the options section:

```
options {
     ..............
     forwarders {9.4.2.1;};
     ..............
};
```

where `9.4.2.1` is the IP address of the machine where queries should be forwarded. This sends unresolved queries to `9.4.2.1` before trying to resolve the query using root name servers (specified in the hints file) or other cached name servers authoritative for or 'closer' to the authoritative name server.

For a name server to *only* use forwarders and not use the root servers, in addition to the forwarders directive, also add the following directive to its BIND 4.9.3 file:

```
options forward-only
```

OR to its BIND 9 conf file:

```
options {
     ..............
     forward only;
};
```

A name server with this option can still answer queries from its cached data. The cache is checked first and if the cache does not contain the answer, the query is sent to the name servers in the forwarders list.

Configuring host resolvers: Name server considerations

If the name server will run on the host being configured, create a loopback file. Specify the loopback address in the *first* name server directive of the resolver configuration file so local clients can access the name server. Refer to "Step 7. Create the loopback file" on page 441 for loopback address considerations.

The BIND 9 name server and BIND 9 DNS utilities (e.g. v9 nslookup and z/OS UNIX dig) use a private resolver that is different from the resolver used by other z/OS UNIX socket programs. The name server has the following functional differences:

- z/OS UNIX nslookup does not use site tables (for example, /etc/hosts) for host name resolution.
- Only the built-in translation table is used for BIND 4.9.3, BIND 9 and all DNS utilities (for example, v9 and v4 nslookup, v9 dig, etc.).

For a complete discussion of resolver configuration files, see *z/OS Communications Server: IP Configuration Reference*.

Configuring host resolvers: onslookup considerations

Programs that query a name server are called resolvers. Because many TCP/IP applications need to query the name server, a set of routines is usually provided for application programmers to perform queries. Under MVS, these routines are available in the TCP/IP application programming interface (API) for each supported language, LE for z/OS UNIX C/C++ Sockets API or z/OS UNIX Assembler Callable Services API.

The v9 nslookup command uses a private resolver that is different from the z/OS UNIX resolver used by other z/OS UNIX socket programs. The v4 nslookup command uses the same resolver as other z/OS UNIX socket programs. The onslookup command has the following functional differences:

- The HFS file, /etc/hosts, is required for host table lookup if name services do not exist. Following is a sample /etc/hosts file:

```
#
# z/OS UNIX Resolver /etc/hosts file on mvss18oe.
#
# The format of this file is:
#
# Internet Address      Hostname   Aliases     # Comments
#
# Items are separated by any number of blanks and/or tabs.  A '#'
# indicates the beginning of a comment; characters up to the end of the
# line are not interpreted by routines which search this file.  Blank
# lines are allowed in this file.

9.24.104.126    mvs18oe mvsoe # z/OS UNIX host
192.168.210.1   mvs18an       # AnyNet MVS host
192.168.210.8   mypcaa        # AnyNet gw host
9.24.104.79     mypc          # A workstation
```

  Note: The presence of /etc/hosts will prevent the z/OS UNIX resolver from accessing *prefix*.HOSTS.SITEINFO and *prefix*.HOSTS.ADDRINFO data sets. The use of /etc/hosts is not recommended unless it is used for purposes other than onslookup.
- Only the built-in translation table is used for both versions of BIND (V4 and V9).

If the z/OS UNIX name server will run on the host being configured, you need to configure the *first* name server (or NsInterAddr) directive in the resolver configuration file as the loopback address (127.0.0.1) or any address in your home list.

Creating the syslog file

If your syslog daemon is not configured, see “Configuring the syslog daemon (syslogd)” on page 101 for information regarding the syslog daemon.

Syslog daemon (syslogd) is a server process that is typically started as one of the first processes in a z/OS UNIX environment. Servers and stack components use syslogd for logging purposes and can also send trace information to syslogd. The named daemon logs messages to the syslog daemon. For BIND 9, specify the *syslog* option in the *channel* phrase of the *logging* statement in the named.conf file in order to use this function. Also for BIND 9, you can direct a category to the *default_syslog* channel. For information about the syslog daemon, see “Configuring the syslog daemon (syslogd)” on page 101.

If you will be using syslogd with BIND 9, refer to “Step 8. For BIND 9 only — configure logging” on page 442 for detailed information.

The name and location of your syslog file is specified in /etc/syslog.conf.

BIND 9 security considerations

Remote Name Daemon Control (rndc)

rndc is a tool that allows the system administrator some degree of control over the name server. The functions available are:

- Reload configuration file and zones.
- Reload the given zone.
- Schedule zone maintenance for the given zone.

- Reload the configuration file and load new zones, but do not reload existing zone files even if they have changed. This is faster than a full reload, when there is a large number of zones, because it avoids the need to examine the modification times of the zone files.
- Write server statistics to the statistics file.
- Toggle query logging.
- Dump the current contents of the cache.
- Stop the server, making sure any recent changes made through dynamic update or IXFR are first saved to the master files of the updated zones.
- Stop the server immediately. Recent changes made through dynamic update or IXFR are not saved to the master files, but will be rolled forward from the journal files when the server is restarted.
- Increment the server's debugging level by one.
- Set the server's debugging level to an explicit value.
- Set the server's debugging level to 0.
- Flush the server's cache.
- Display status of the server.

For more detail, refer to *z/OS Communications Server: IP System Administrator's Commands*, the rndc man page, the rndc.conf man page, and the rndc-confgen man page.

A configuration file is required, since all communication with the server is authenticated with digital signatures that rely on a shared secret, and there is no way to provide that secret other than with a configuration file. The default location for the rndc configuration file is /etc/rndc.conf, but an alternate location can be specified with the -c option. If the configuration file is not found, rndc will also look in /etc/rndc.key. The rndc.key file is generated by running rndc-confgen -a.

The format of the configuration file is similar to that of named.conf, but limited to only four statements:

- options
- key
- server
- include

These statements are what associate the secret keys to the servers with which they are meant to be shared. The order of statements is not significant.

The options statement has three clauses:

- default-server
- default-key
- default-port

The default-server clause takes a host name or address argument and represents the server that will be contacted if no -s option is provided on the command line. The default-key clause takes the key name as its argument, as defined by a key statement. The default-port clause specifies the port to which rndc should connect if no port is given on the command line or in a server statement.

The key statement names a key with its string argument. The string is required by the server to be a valid domain name, though it need not actually be hierarchical; thus, a string like ″rndc_key″ is a valid name. The key statement has two clauses:

- algorithm
- secret

While the configuration parser will accept any string as the argument to algorithm, currently only the string ″hmac-md5″ has any meaning. The secret is a base-64 encoded string.

Since the tool may be used remotely, rndc and the name server must communicate using digital transaction signatures (TSIG). Therefore, rndc and the name server must be configured with a *shared-secret*. There are two ways to configure a shared-secret key. One way is to use the /etc/rndc.key file generated by the rndc-confgen -a command. This file is shared between the name server and the rndc utility. The other way is to generate a shared-secret TSIG key with the HMAC-MD5 algorithm using the dnssec-keygen utility. The key must be configured in the name server under the *controls* section, and in the rndc.conf file on the key clause.

The server statement uses the key clause to associate the server with a key. The argument to the server statement is a host name or address (addresses must appear in double quotation marks). The argument to the key clause is the name of the key as defined by the key statement. The port clause can be used to specify the port to which rndc should connect on the given server.

The include statement can be used to insert the contents of another file within the rndc configuration file (for example, to include the contents of a file that contains sensitive key information).

A sample minimal configuration file is as follows:

```
key rndc_key {
     algorithm "hmac-md5";
     secret "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};
options {
     default-server localhost;
     default-key    rndc_key;
};
```

This file, if installed as /etc/rndc.conf, would allow the rndc reload command to connect to 127.0.0.1 port 953 and cause the nameserver to reload, if a nameserver on the local machine were running with the following controls statements and it had an identical key statement for rndc_key:

```
controls {
        inet 127.0.0.1 allow { localhost; } keys { rndc_key; };
};
```

Running the rndc-confgen program will conveniently create a rndc.conf file for you, and also display the corresponding controls statement that you need to add to named.conf. Alternatively, you can run rndc-confgen -a to set up a rndc.key file and not modify named.conf at all.

Access Control Lists

Access Control Lists (ACLs), are address match lists that you can set up and nickname for future use in allow-query , allow-recursion , blackhole , allow-transfer , etc. Using ACLs allows you to have finer control over who can access your nameserver, without cluttering up your config files with huge lists of IP addresses. It is a good idea to use ACLs, and to control access to your server. Limiting access to

your server by outside parties can help prevent spoofing and DoS (Denial of Service) attacks against your server. Here is an example of how to properly apply ACLs:

```
// Set up an ACL named "bogusnets" that will block RFC1918 space,
// which is commonly used in spoofing attacks.

acl bogusnets { 0.0.0.0/8; 1.0.0.0/8; 2.0.0.0/8; 192.0.2.0/24; 224.0.0.0/3;
10.0.0.0/8; 172.16.0.0/12; 192.168.0.0/16; };

// Set up an ACL called our-nets. Replace this with the real IP numbers.

acl our-nets { x.x.x.x/24; x.x.x.x/21; };

options {
  ...
  ...
  allow-query { our-nets; };
  allow-recursion { our-nets; };
  ...
  blackhole { bogusnets; };
  ...
};
zone "example.com" {
  type master;
  file "m/example.com";
  allow-query { any; };
};
```

This allows non-recursive queries for example.com from inside and outside nets, except from bogusnets, and allows recursive queries from our-nets to outside nets unless *recursion no;* is also specified in the configuration file.

For more information on how to use ACLs to protect your server, see the AUSCERT advisory at: ftp://ftp.auscert.org.au/pub/auscert/advisory/AL-1999.004.dns_dos

chroot and setuid

It is possible to run BIND in a chrooted environment (**chroot()**) by specifying the *-t* option. This can help improve system security by placing BIND in a sandbox, which will limit the damage done if a server is compromised.

Another useful feature is the ability to run the daemon as a nonprivileged user (-u user). We suggest running as a nonprivileged user when using the **chroot** feature.

Here is an example command line to load BIND in a **chroot()** sandbox, **/var/named**, and to run **named setuid** to user 202:

```
<>/usr/local/bin/named -u 202 -t /var/named
```

The chroot environment: In order for a **chroot**() environment to work properly in a particular directory (for example, /var/named), you will need to set up an environment that includes everything BIND needs to run. From BIND's point of view, /var/named is the root of the filesystem. You will need /dev/null, and any library directories and files that BIND needs to run on your system.

Using the setuid function: Prior to running the **named** daemon, use the **touch** utility (to change file access and modification times) or the **chown** utility (to set the user id and/or group id) on files to which you want BIND to write.

Dynamic update security

Access to the dynamic update facility should be strictly limited. For these reasons, we strongly recommend that updates be cryptographically authenticated by means

of transaction signatures (TSIG). That is, the **allow-update** option should list only TSIG key names, not IP addresses or network prefixes. Alternatively, the **update-policy** option can be used.

Some sites choose to keep all dynamically updated DNS data in a subdomain and delegate that subdomain to a separate zone. This way, the top-level zone containing critical data such as the IP addresses of public web and mail servers need not allow dynamic update at all.

Special considerations when using Dynamic VIPA

If you run a name server on a host that is using Dynamic VIPA (DVIPA), you may be required to do some additional configuration. Name servers running on a host using DVIPA need to BIND the UDP port that the name server listens on (usually 53) to the DVIPA, if you wish DNS to make use of the DVIPA. This can be done by using the BIND option on the UDP PORT statement in the TCPIP.PROFILE. If you do BIND the DNS UDP port to the DVIPA, then all references to that name server must use the DVIPA whether those references are from other name servers or from resolvers.

References to a name server could occur in a number of places, and should be changed to use the DVIPA if BINDing a DNS UDP port to the DVIPA. This list is not exhaustive, but is intended to aid you for some of the most common cases. In general, you may need to change any place that references a name server by its IP address when using DVIPAs.

Task	Location
Delegating a DNS subdomain to a name server running on a host using DVIPA	The 'A' record in the glue records for the delegated (child) name server in the domain data file of the delegating (parent) name server
Designating a slave (secondary) name server when master (primary) name server is running on a host using DVIPA	The 'secondary' statement in the boot file of the secondary (slave) name server for BIND 4.9.3 or the 'masters' option of the 'zone' statement for BIND 9
Configuring resolvers	'NSINTERADDR' or 'nameserver' directive of the resolver configuration file
Using a name server as the target of other forwarding name servers when the target name server resides on a host using DVIPA	'forwarders' directive in the boot file of the forwarding name servers for BIND 4.9.3 or the 'forwarders{}' option of the 'options{}' statement for BIND 9
Using a name server as an intranet root name server when the root name server is running on a host using DVIPA	'A' record of the intranet root name server in the hints file on all name servers within the intranet

Dynamic primary DNS movement using Dynamic VIPA

To use DNS along with DVIPA takeover functionality to provide a high availability environment, perform the following steps:

1. Define a distributed DVIPA with the VIPADISTribute statement, specifying a port on the PORT parameter that will not be used by the Sysplex Distributor (to prevent undesired distribution of connections). DESTIP should be the DXCF addresses of all the members that will be backup servers. For example:

```
VIPADYNAMIC
VIPADEFINE     255.255.255.192 10.134.61.190
VIPADISTRIBUTE 10.134.61.190 PORT 8081 DESTIP ALL
ENDVIPADYNAMIC
```

2. Define VIPABACKUPs on all the members that will be backup servers. For example:

```
VIPADYNAMIC
VIPABACKUP  50  10.134.61.190
ENDVIPADYNAMIC
```

3. On both the Distributor and all the targets, define the following statements.

   Note: The following example applies to the BIND 4.9.3 name server only.

```
PORT
⋮
53 TCP NAMED2 BIND 10.134.61.190
53 UDP NAMED2 BIND 10.134.61.190
```

4. Update /etc/resolv.conf and TCPDATA on all SYSPLEX members to point to that address. For example:

```
⋮
NSINTERADDR  10.134.61.190
⋮
```

5. Update all of the glue records (NS records and their corresponding A records), in the sysplex name servers and the sysplex parent's name servers that point to the sysplex name server, to use the dynamic VIPA.

After doing the above configuration, stack termination will cause the DVIPA to be taken over, making DNS on the new DVIPA owning stack reachable after route convergence completes (OMPROUTE recommended).

Querying name servers

This section describes how to use the nslookup command to query the name server. onslookup is an alias of nslookup in the z/OS UNIX environment.

Notes:

1. The z/OS UNIX nslookup command runs only from the z/OS shell. The nslookup command can query the name server from TSO or the z/OS shell. However, only the legacy TSO version of NSLOOKUP is available from TSO. Refer to *z/OS Communications Server: IP System Administrator's Commands* for detailed information.
2. The host and dig commands are another way to query name servers from the z/OS shell. See the *z/OS Communications Server: IP User's Guide and Commands* for a list of host commands.

nslookup command

The z/OS UNIX nslookup and TSO NSLOOKUP commands can be used to query the name server to perform the following tasks:

- Identifying the location of name servers
- Examining the contents of a name server database
- Establishing the accessibility of name servers

Note: sortlist is not supported by nslookup

The z/OS UNIX nslookup and TSO NSLOOKUP commands have two modes of operation: interactive mode and command mode. nslookup also has two versions in

z/OS UNIX: v4 and v9, where v4 gives the legacy z/OS UNIX (o)nslookup function, and v9 gives the BIND 9 version of nslookup. In either mode, the address of the default name server comes from the resolver configuration data. In the sample data below, the default domain is raleigh.ibm.com, and the default name server is at 9.37.34.149. If that name server fails to respond, the one at 9.37.34.7 is used.

```
domain   raleigh.ibm.com
nameserver   9.37.34.149
nameserver   9.37.34.7
```

Entering the interactive mode

Interactive mode can be used to repetitively query one or more name servers for information about various hosts and domains, to display that information on the console, and, in some cases, to write response data to a file.

You can enter the interactive mode under the following conditions only:

- No arguments are supplied on command invocation or the –v option is specified; the default name server is used.
- The first argument is a hyphen, and the second argument is the host name or Internet address of a name server.

For a complete description of the z/OS UNIX and TSO nslookup interactive modes, refer to *z/OS Communications Server: IP User's Guide and Commands*.

Entering the command line mode

The command line mode displays or stores the output from the query supplied as part of the invocation string and then exits.

To enter the command line mode, provide a complete query with the z/OS UNIX nslookup command invocation string.

For a complete description of the z/OS UNIX and TSO nslookup command line modes, refer to *z/OS Communications Server: IP System Administrator's Commands*.

nslookup configuration

nslookup for BIND 4.9.3: The configuration options of z/OS UNIX nslookup determine the operation and results of the name server queries. The values for z/OS UNIX nslookup options can be specified in more than one location, as shown in Table 19 on page 465. Values specified as z/OS UNIX nslookup command options have priority over values specified in the `.onslookuprc` file (for nslookup version 4), which have priority over the value specified by the environment variable, and so on. For example, the value specified by the `querytype` option in the z/OS UNIX nslookup command has priority over the value specified by the `querytype` option in the `.onslookuprc` file.

The letters beside some settings indicate that the *terms* are functionally equivalent. For example, the term `domain` (the letter ″A″) is functionally equivalent to the terms `DomainOrigin` and `LOCALDOMAIN`. If two functionally equivalent settings are listed in the same file, the one listed last has priority. For example, if domain and DomainOrigin are both listed in the Resolver configuration data set, and domain is listed first, the value specified by DomainOrigin has priority.

The numeric column headings in Table 19 on page 465 correspond to the following:

1 z/OS UNIX nslookup command options. Refer to *z/OS Communications Server: IP User's Guide and Commands* for more details about the z/OS UNIX nslookup command.

2 .onslookuprc file in the home directory. Refer to *z/OS Communications Server: IP User's Guide and Commands* for more details.

3 Environment variable (set by the z/OS UNIX command ″export LOCALDOMAIN=domain_origin″).

4 `Resolver configuration data set` Refer to *z/OS Communications Server: IP Configuration Reference* for more details.

For example, column 1 lists z/OS UNIX nslookup options and column 2 lists options you can set in the `.onslookuprc` file.

Table 19. Settings that affect nslookup operation

Settings	**1**	**2**	**3**	**4**
All	x	x		
Class	x	x		
no[d2]	x	x		
[no]debug	x	x		
[no]defname	x	x		
domain (A)	x	x		x
[no]ignoretc	x	x		
port (B)	x	x		
querytype	x	x		
[no]recurse	x	x		
retry (C)	x	x		
root	x	x		
[no]search	x	x		
srchlist (D)	x	x		
timeout (E)	x	x		
[no]vc (F)	x	x		
search (D)				x
nameserver (G)				x
sortlist				x
options debug				x
options ndots				x
DomainOrigin (A)				x
NsInterAdd (G)				x
NsPortAddr (B)				x
ResolveVia (F)				x
ResolverTimeout (E)				x
ResolverUdpRetries (C)				x
LOCALDOMAIN (A)			x	

nslookup for BIND 9: There are fewer ways to configure BIND 9 (v9) nslookup in comparison to BIND 4.9.3 (v4) nslookup. There are only two places to specify v9 nslookup options: as command options, or from the resolver configuration data set. Only a few of the options may be set in the resolver configuration data set. The command options always have precedence over any option configured in the resolver configuration data set.

Only the following options can be specified in the resolver configuration data set for v9 nslookup:

- nameserver/nsinteraddr
- options ndots: *n*
- search
- domain/domainorigin

Programs that query a name server are called resolvers. Refer to "Understanding resolvers" on page 12 for more detailed information. Because many TCP/IP applications need to query the name server, a set of routines is usually provided for application programmers to perform queries. Under MVS, these routines are available in the TCP/IP application programming interface (API) for each supported language or LE for z/OS UNIX Sockets API.

The z/OS UNIX v4 and v9 nslookup commands use a private resolver that is different from the resolver used by other z/OS UNIX socket programs. The z/OS UNIX nslookup command has the following functional differences:

- z/OS UNIX nslookup does not use SiteTables (for example, /etc/hosts) for host name resolution.
- Only the built-in translation table is used for v4 and v9 nslookup.
- z/OS UNIX v4 nslookup uses the LOCALDOMAIN environment variable, whereas no other resolvers use this.

For a complete discussion of resolver configuration files, see Chapter 1, "Configuration overview" on page 3.

If the name server will run on the host being configured, you need to configure the *first* name server (or NsInterAddr) directive in the resolver configuration file as the loopback address (127.0.0.1 or any address in your home list).If any VIPA addresses are used with the NsInterAddr statement, ensure that IPCONFIG SOURCEVIPA is coded in PROFILE.TCPIP. If it is not, UDP packets returned from the VIPA address will have the physical interface address as the destination address instead of the VIPA address that it sent. The UDP packet will be discarded when it is received because the addresses do not match.

Diagnosing problems

This section describes the following methods for diagnosing problems:

- Checking messages on the operators console
- Checking the syslog messages
- Using name server signals
- Using rndc to diagnose BIND 9 problems
- Checking name server logging files to diagnose BIND 9
- Using nslookup program
- Using the dig command

These methods are discussed below. In addition to these methods, diagnosing problems for a dynamic zone can be done with nsupdate. This allows debugging of problems between DHCP and the dynamic zone in a more controlled manner.

For DNS configuration firewall considerations, refer to “Split DNS” on page 471, and the latest edition of *DNS and BIND* by Paul Albitz and Cricket Liu (O’Reilly & Associates, Inc.).

Checking messages sent to the operators console

Messages displayed on the operators console indicate the status of your DNS. Messages fall into the following categories:

- Name server initialization
- Name server initialization failure
- Name server initialization complete
- Name server termination
- Assertion failures (unexpected errors) (v9 only)

Regularly check console messages to identify problems.

Checking the syslog messages

Error messages may also be displayed in the syslog output file, which is pointed to by the syslog configuration file. (`/etc/syslog.conf` is the default configuration file.) For BIND 9, refer to “Step 8. For BIND 9 only — configure logging” on page 442. For the BIND 9 name server, initial startup messages go to syslog, later messages will be directed to other defined or default logs according to logging statements found or implied in the configuration file. For descriptions of the syslog file and the syslog daemon, see “Configuring the syslog daemon (syslogd)” on page 101.

Using name server signals to diagnose BIND 4.9.3 DNS problems

You can use name server signals to send messages to a BIND 4.9.3 or a BIND 9 DNS. Note that some of the signals may have different consequences if a signal is sent to a BIND 4.9.3 name server instead of a BIND 9 nameserver. These signals control various functions that can be used to diagnose problems.

Diagnostic functions include the following:

- Enabling and disabling debug message logging
- Dumping the contents of the name server database
- Getting short status
- Logging queries

For an explanation of the format of the dumped database or the name server statistics file that can be generated with these signals, refer to publications such as *DNS and Bind* by Albitz and Liu.

The issuing of a signal is done by using the z/OS UNIX kill command and also involves specifying the process ID. The BIND 4.9.3 name server always stores its process ID in the file, /etc/named.pid.

Using name server signals to diagnose BIND 9 DNS problems

You can use name server signals to send messages to a BIND 4.9.3 or a BIND 9 DNS name server. Note that some of the signals may have different consequences

if a signal is sent to a BIND 4.9.3 name server instead of a BIND 9 name server. These signals control various functions that can be used to diagnose problems.

The BIND 9 name server relies on a start option, rndc, or the configuration file to define and alter the debug level. You can change the logging options in named.conf to gather more information, and then issue the SIGHUP signal or 'rndc reload' to have the new logging options take effect. However, the preferred method is by using 'rndc trace level'.

For an explanation of the format of the dumped database or the name server statistics file that can be generated with these signals, refer to publications like *DNS and Bind* by Albitz and Liu.

Signals are issued with the z/OS UNIX kill command using the name server process ID as a parameter. The file where the BIND 9 process ID is stored is determined by the user. The file name is specified by the 'pid-file' option in the named.conf file. Refer to *z/OS Communications Server: IP Configuration Reference* for further details.

Using rndc to diagnose BIND 9 problems

The rndc utility can be used to provide a variety of functions that can be helpful in debugging name server problems. For example, the name server's cache can be viewed using the dumpdb parameter and debug trace can be turned on or off using the trace parameter. If you suspect your cache is corrupted, you can flush the name server's cache with the flush parameter. For more information, see *z/OS Communications Server: IP System Administrator's Commands*.

Checking name server logging files to diagnose BIND 9

Error, debug and informational messages can be written to the name server's logging files. Refer to "Step 8. For BIND 9 only — configure logging" on page 442 for more information.

Using nslookup to diagnose problems

The z/OS UNIX nslookup program lets you query other name servers with the same query packet another name server would use. This is helpful in diagnosing lookup problems in TCP/IP.

It is recommended that you use z/OS UNIX or TSO nslookup with each NsInterAddr used in TCPIP.DATA to ensure you receive the expected results. Some name server clients, on other platforms, may require the address you specify for the name server to match the source IP address in the response from the name server. For example, if a static VIPA address is specified as the address of the name server, and IPCONFIG SOURCEVIPA is not specified in PROFILE.TCPIP, then nslookup on some platforms will discard the returned packet because it will have the destination address of the physical interface instead of the VIPA interface. If you wish to specify a Dynamic VIPA (DVIPA) as the address of the name server, then the name server must BIND the UDP port to the DVIPA. Refer to *z/OS Communications Server: IP Configuration Reference* for information on how to specify the BIND parameter on the PORT statement in the TCPIP.PROFILE.

Debugging is available at different levels for BIND 9 and 4.9.3. The commands are similar but the output differs between BIND 9 and BIND 4.9.3. Refer to the *z/OS Communications Server: IP System Administrator's Commands* for more detailed information on the z/OS UNIX nslookup command.

Using dig to diagnose problems

The dig command is an alternate and recommended choice for resource record lookup. The dig response format is similar to the resource record (RR) definitions in master zone files. The dig command will not attempt a reverse lookup on the address provided for the server, which sometimes makes nslookup fail initialization if reverse lookup fails. The dig command offers flexible options, including toggling flags for checking response authority or authentication. It is the only v9 lookup utility that can list the complete contents of a zone. This can be accomplished with the -t AXFR option.

Advanced BIND 9 name server topics

Multiple TCP/IP stack (common INET) considerations

The BIND 9 name server is a *generic* server which does not have stack affinity. This is in contrast to the BIND 4.9.3 name server which does have stack affinity. This has certain implications.

- If you wish to run multiple BIND 9 name servers, you must divide the interfaces between the name servers with the **listen-on** named.conf file option. For example, you may want one stack serviced by one BIND 9 name server, and a second stack serviced by a second BIND 9 name server. Each name server would contain only the IP addresses of the assigned stack in its **listen-on** option.
- Any time a stack is brought up or a stack is brought down, the name server will essentially restart. On the MVS console, you will see the NAMED Exiting, NAMED Starting, and NAMED Running messages (messages EZZ9096I, EZZ9095I, and EZZ9130I).
- Be aware, that once you start a TCP/IP stack, all of the adapters may not be active immediately, and therefore will not be usable by the name server immediately. When the name server is started manually or restarted automatically by stack bring up or bring down, it immediately queries the available TCP/IP stacks for active adapters. Often times, it will take some time for all of the adapters to become active (this is independent of the name server). The name server will re-query the stacks every minute, by default, for any changes in the active/inactive status of adapters and then make use of them once they are active. The one minute interval can be lengthened by the **interface-interval** named.conf file option if desired, but this is not recommended.
- By default, the name server will unpredictably choose one adapter from any of the active stacks to use when it must communicate with other name servers. If some adapters do not have the capability to route into the network, you may see unpredictable results on name server queries. This unpredictable behavior can be eliminated by making use of the **query-source** option in the named.conf file. The **query-source** option should specify an adapter address that will always have network routing capability. The **query-source** option then places a dependency on the stack that owns that address to be active. If the owning TCP/IP stack of the **query-source** option address is taken down, the name server will end, since it will no longer have a way to communicate with the network, and thus, other name servers.
- BIND 9 is restricted to listening on all IPv6 interfaces or none of them. Therefore, if you want to run multiple BIND 9 name servers, you need to choose one of them to answer all IPv6 queries. Use the listen-on-v6 named.conf option with the value of *any;* to get BIND 9 to listen on your IPv6 interfaces.

Dynamic update

Dynamic update is the term used for the ability under certain specified conditions to add, modify or delete records or RRsets in the master zone files. Dynamic update is fully described in RFC 2136.

Dynamic update is enabled on a zone-by-zone basis, by including an **allow-update** or **update-policy** clause in the **zone** statement. Preferably, use TSIG security between the nsupdate utility and the targeted name server. BIND 9 name server configuration processing messages will remind you when nsupdate authorization is only based on client IP address.

Updating of secure zones (zones using DNSSEC) is modelled after the simple-secure-update proposal, a work in progress in the DNS Extensions working group of the IETF. (See http://www.ietf.org/html.charters/dnsext-charter.html for information about the DNS Extensions working group.) SIG and NXT records affected by updates are automatically regenerated by the server using an online zone key. Update authorization is based on transaction signatures and an explicit server policy. On z/OS, dynamic DNSSEC zones should use the RSA encryption algorithm when creating the zone key, or they can use the DSA algorithm if the 'random-device' option is also specified in the named.conf file. Non-dynamic DNSSEC zones can use any other supported encryption algorithm.

The zone files of dynamic zones must not be edited by hand. If the zone file of a dynamic zone is edited by hand, corrupt .jnl files can result and all changes not written to the zone file may be lost.The zone file on disk at any given time may not contain the latest changes performed by dynamic update. The zone file is written to disk only periodically, and changes that have occurred since the zone file was last written to disk are stored only in the zone's journal (.jnl) file. Depending on signal or rndc stop options, BIND 9 name server may or may not update the zone file. Therefore, editing the zone file manually is unsafe even when the server has been shut down.

Incremental zone transfers (IXFR)

The incremental zone transfer (IXFR) protocol is a way for slave servers to transfer only changed data, instead of having to transfer the entire zone. The IXFR protocol is documented in RFC 1995.

When acting as a master server, BIND 9 supports IXFR for those zones where the necessary change history information is available. These include master zones maintained by dynamic update and slave zones (to transfer to other slave servers) whose data was obtained by IXFR, but not manually maintained master zones or slave zones obtained by performing a full zone transfer (AXFR).

When acting as a slave server, after the initial full zone transfer, BIND 9 will request IXFR by default when notified of a change by the zone master server. IXFR updates are applied to the slave zone database and also kept in a journal file (*.jnl) associated with any existing backup file for the slave zone.

Master and slave servers can each have IXFR globally or partially disabled through the use of the *provide-ixfr* and *request-ixfr* options under the general options or the *server* statements of the BIND 9 configuration file.

Split DNS

Setting up different views, or visibility, of DNS space to internal and external resolvers is usually referred to as a Split DNS setup. There are several reasons an organization would want to set up its DNS this way.

One common reason for setting up a DNS system this way is to hide *internal* DNS information from *external* clients on the Internet. There is some debate as to whether or not this is actually useful. Internal DNS information leaks out in many ways (via e-mail headers, for example) and most savvy attackers can find the information they need using other means.

Another common reason for setting up a Split DNS system is to allow internal networks that are behind filters or in RFC 1918 space (reserved IP space, as documented in RFC 1918) to resolve DNS on the Internet. Split DNS can also be used to allow mail from outside back in to the internal network.

Here is an example of a split DNS setup:

A company named Example, Inc. (example.com) has several corporate sites that have an internal network with reserved Internet Protocol (IP) space and an external demilitarized zone (DMZ), or *outside* section of a network, that is available to the public.

Example, Inc. wants its internal clients to be able to resolve external hostnames and to exchange mail with people on the outside. The company also wants its internal resolvers to have access to certain internal-only zones that are not available at all outside of the internal network.

In order to accomplish this, the company will set up two sets of nameservers. One set will be on the inside network (in the reserved IP space) and the other set will be on bastion hosts, which are *proxy* hosts that can talk to both sides of its network, in the DMZ.

The internal servers will be configured to forward all queries, except queries for site1.internal, site2.internal, site1.example.com, and site2.example.com, to the servers in the DMZ. These internal servers will have complete sets of information for site1.example.com, site2.example.com, site1.internal, and site2.internal.

To protect the site1.internal and site2.internal domains, the internal nameservers must be configured to disallow all queries to these domains from any external hosts, including the bastion hosts.

The external servers, which are on the bastion hosts, will be configured to serve the *public* version of the site1 and site2.example.com zones. This could include things such as the host records for public servers (www.example.com and ftp.example.com), and mail exchange (MX) records (a.mx.example.com and b.mx.example.com).

In addition, the public site1 and site2.example.com zones should have special MX records that contain wildcard (*) records pointing to the bastion hosts. This is needed because external mail servers do not have any other way of looking up how to deliver mail to those internal hosts. With the wildcard records, the mail will be delivered to the bastion host, which can then forward it on to internal hosts.

Here's an example of a wildcard MX record:

```
*   IN MX 10 external1.example.com.
```

Now that they accept mail on behalf of anything in the internal network, the bastion hosts will need to know how to deliver mail to internal hosts. In order for this to work properly, the resolvers on the bastion hosts will need to be configured to point to the internal nameservers for DNS resolution. Queries for internal hostnames will be answered by the internal servers, and queries for external hostnames will be forwarded back out to the DNS servers on the bastion hosts. In order for all this to work properly, internal clients will need to be configured to query only the internal nameservers for DNS queries. This could also be enforced via selective filtering on the network.

If everything has been set properly, Example, Inc.'s internal clients will now be able to:

- Look up any hostnames in the site1 and site2.example.com zones.
- Look up any hostnames in the site1.internal and site2.internal domains.
- Look up any hostnames on the Internet.
- Exchange mail with internal and external people.

Hosts on the Internet will be able to:

- Look up any hostnames in the site1 and site2.example.com zones.
- Exchange mail with anyone in the site1 and site2.example.com zones.

Here is an example configuration for the setup we just described above. Note that this is only configuration information; for information on how to configure your zone files, see "Step 5. Create the domain data files (master name server only)" on page 433.

Internal DNS server config:

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };

acl externals { bastion-ips-go-here; };

options {
    ...
    ...
    forward only;
    forwarders {     // forward to external servers
     bastion-ips-go-here;
    };
    allow-transfer { none; };   // sample allow-transfer (no one)
    allow-query { internals; externals; }; // restrict query access
    allow-recursion { internals; };  // restrict recursion
    ...
    ...
};

zone "site1.example.com" {
  type master;
  file "m/site1.example.com";
  forwarders { };     // do normal iterative
      // resolution (do not forward)
  allow-query { internals; externals; };
  allow-transfer { internals; };
};

zone "site2.example.com" {
  type slave;
```

```
  file "s/site2.example.com";
  masters { 172.16.72.3; };
  forwarders { };
  allow-query { internals; externals; };
  allow-transfer { internals; };
};

zone "site1.internal" {
  type master;
  file "m/site1.internal";
  forwarders { };
  allow-query { internals; };
  allow-transfer { internals; }
};

zone "site2.internal" {
  type slave;
  file "s/site2.internal";
  masters { 172.16.72.3; };
  forwarders { };
  allow-query { internals };
  allow-transfer { internals; }
};
```

External (bastion host) DNS server config:

```
acl internals { 172.16.72.0/24; 192.168.1.0/24; };

acl externals { bastion-ips-go-here; };

options {
  ...
  ...
  allow-transfer { none; };   // sample allow-transfer (no one)
  allow-query { internals; externals; }; // restrict query access
  allow-recursion { internals; externals; }; // restrict recursion
  ...
  ...
};

zone "site1.example.com" {
  type master;
  file "m/site1.foo.com";
  allow-query { any; };
  allow-transfer { internals; externals; };
};

zone "site2.example.com" {
  type slave;
  file "s/site2.foo.com";
  masters { another_bastion_host_maybe; };
  allow-query { any; };
  allow-transfer { internals; externals; }
};
```

In the resolv.conf (or equivalent) on the bastion host(s):

```
search ...
nameserver 172.16.72.2
nameserver 172.16.72.3
nameserver 172.16.72.4
```

Implementing split DNS with views

The view statement is a powerful new feature of BIND 9 that lets a name server answer a DNS query differently depending upon who is asking. It is particularly useful for implementing split DNS setups without having to run multiple servers.

Each view statement defines a view of the DNS namespace that will be seen by a subset of clients. A client matches a view if its source IP address matches the address_match_list of the view's match-clients clause and its destination IP address matches the address_match_list of the view's match-destinations clause. If not specified, both match-clients and match-destinations default to matching all addresses. A view can also be specified as match-recursive-only, which means that only recursive requests from matching clients will match that view. The order of the view statements is significant; A client request will be resolved in the context of the first view that it matches.

Zones defined within a view statement will only be accessible to clients that match the view. By defining a zone of the same name in multiple views, different zone data can be given to different clients (for example, internal and external clients in a split DNS setup.

Many of the options given in the options statement can also be used within a view statement, and then apply only when resolving queries with that view. When no view-specific value is given, the value in the options statement is used as a default. Also, zone options can have default values specified in the view statement. These view-specific defaults take precedence over those in the options statement.

Views are class specific. If no class is given, class IN is assumed. Note that all non-IN views must contain a hint zone, since only the IN class has compiled-in default hints.

If there are no view statements in the configuration file, a default view that matches any client is automatically created in class IN, and any zone statements specified on the top level of the configuration file are considered to be part of this default view. If any explicit view statements are present, all zone statements must occur inside view statements.

Following is an example of a typical split DNS setup implemented using view statements:

```
view "internal" {
               // This should match our internal networks.
      match-clients { 10.0.0.0/8; };
               // Provide recursive service to internal clients only.
      recursion yes;
               // Provide a complete view of the example.com zone
               // including addresses of internal hosts.
      zone "example.com" {
            type master;
            file "example-internal.db";
      };
};
view "external" {
      match-clients { any; };
               // Refuse recursive service to external clients.
      recursion no;
               // Provide a restricted view of the example.com zone
               // containing only publicly accessible hosts.
      zone "example.com" {
           type master;
           file "example-external.db";
      };
};
```

TSIG

This is a short guide to setting up Transaction SIGnatures (TSIG) based transaction security in BIND. It describes changes to the configuration file as well as what changes are required for different features, including the process of creating transaction keys and using transaction signatures with BIND.

BIND primarily supports TSIG for server to server communication. This includes zone transfer, notify, and recursive query messages. Resolvers on other platforms which are based on newer versions of BIND 8 have limited support for TSIG.

TSIG might be most useful for dynamic update. A master server for a dynamic zone should use access control to control updates, but IP-based access control is insufficient. Key-based access control is far superior. The **nsupdate** program supports TSIG via the -k and -y command line options.

Generate shared keys for each pair of hosts

A shared secret is generated to be shared between host1 and host2. An arbitrary key name is chosen: ″host1-host2.″. The key name must be the same on both hosts.

Automatic generation: The following command will generate a 128–bit (16–byte) HMAC-MD5 key as described above. Longer keys are better, but shorter keys are easier to read. Note that the maximum key length is 512 bits; keys longer than that will be digested with MD5 to produce a 128 bit key.

```
dnssec-keygen -a hmac-md5 -b 128 -n HOST host1-host2.
```

The key is in the file Khost1-host2.+157+00000.private. Nothing directly uses this file, but the base-64 encoded string following ″Key:″ can be extracted from the file and used as a shared secret:

```
Key: La/E5CjG9O+os1jq0a2jdA==
```

The string ″La/E5CjG9O+os1jq0a2jdA==″ can be used as the shared secret.

Manual generation: The shared secret is simply a random sequence of bits, encoded in base-64. Most EBCDIC strings are valid base-64 strings (assuming the length is a multiple of 4 and only valid characters are used), so the shared secret can be manually generated. Also, a known string can be run through *mmencode* on another platform (such as Linux), or through a similar program, to generate base-64 encoded data.

Copying the shared secret to both machines

This is beyond the scope of DNS. A secure transport mechanism should be used. This could be secure FTP, ssh, telephone, etc.

Informing the servers of the key's existence

Imagine *host1* and *host2* are both servers. The following is added to each server's named.conf file:

```
key host1-host2. {
  algorithm hmac-md5;
  secret "La/E5CjG9O+os1jq0a2jdA==";
};
```

The algorithm, hmac-md5, is the only one supported by BIND. The secret is the one generated above. Since this is a secret, it is recommended that either named.conf be non-world readable, or the key directive be added to a non-world readable file that is included by named.conf.

At this point, the key is recognized. This means that if the server receives a message signed by this key, it can verify the signature. If the signature succeeds, the response is signed by the same key.

Instructing the server to use the key

Since keys are shared between two hosts only, the server must be told when keys are to be used. The following is added to the named.conf file for host1, if the IP address of host2 is 10.1.2.3:

```
server 10.1.2.3 {
  keys { host1-host2. ;};
};
```

Multiple keys may be present, but only the first is used. This directive does not contain any secrets, so it may be in a world-readable file.

If host1 sends a message that is a request to that address, the message will be signed with the specified key. host1 will expect any responses to signed messages to be signed with the same key. A similar statement must be present in host2's configuration file (with host1's address) for host2 to sign request messages to host1.

TSIG key based access control

BIND allows IP addresses and ranges to be specified in ACL definitions, and in access control directives such as allow-query, allow-transfer, and allow-update. This has been extended to allow TSIG keys also. The above key would be denoted **key host1-host2**. An example of an allow-update directive would be:

```
allow-update { key host1-host2. ;};
```

This allows dynamic updates to succeed only if the request was signed by a key named ″**host1-host2**.″.

Errors

The processing of TSIG signed messages can result in several errors. If a signed message is sent to a non-TSIG aware server, a FORMERR will be returned, since the server will not understand the record. This is a result of misconfiguration, since the server must be explicitly configured to send a TSIG signed message to a specific server.

If a TSIG aware server receives a message signed by an unknown key, the response will be unsigned with the TSIG extended error code set to BADKEY. If a TSIG aware server receives a message with a signature that does not validate, the response will be unsigned with the TSIG extended error code set to BADSIG. If a TSIG aware server receives a message with a time outside of the allowed range, the response will be signed with the TSIG extended error code set to BADTIME, and the time values will be adjusted so that the response can be successfully verified. In any of these cases, the message's rcode is set to NOTAUTH.

DNSSEC

Cryptographic authentication of DNS information is possible through the DNS Security (DNSSEC) extensions, defined in RFC 2535. This section describes the creation and use of DNSSEC signed zones.

The set of *dnssec-* tools rely on a /dev/random device for the entropy it needs to generate cryptographically strong keys. If RSA keys are used, only *dnssec-keygen* requires random data. z/OS UNIX does not include such a device, but the tools

provide alternate methods of providing them with random data. The user can specify a file containing random data or can provide random data via the keyboard. To specify a file, use the **-r** *random data file* option on the tool command line. The *dnssec-* tools use the timing between keystrokes as the source of entropy. As such, TN3270 terminal emulation is not the ideal interface. Setting up a VT100 terminal session is a better solution. Refer to the "Configuring the z/OS UNIX Telnet server (otelnetd)" on page 374 for more information on setting up otelnetd.

To set up a DNSSEC secure zone, there are a series of steps which must be followed. z/OS ships with several tools that are used in this process, which are explained in more detail below. In all cases, the -h option prints a full list of parameters. Note that the keyset and signedkey files created by some DNSSEC tools must be put in the name server working directory before another DNSSEC tool is used for signing a master zone file.

There must also be communication with the administrators of the parent and/or child zone to transmit keys and signatures. A zone's security status must be indicated by the parent zone for a DNSSEC capable resolver to trust its data.

For other servers to trust data in this zone, they must be statically configured with either this zone's zone key or the zone key of another zone above this one in the DNS tree, using the *trusted-keys* statement in the configuration file.

Generating keys

The **dnssec-keygen** program is used to generate keys.

A secure zone must contain one or more zone keys. The zone keys will sign all other records in the zone, as well as the zone keys of any secure delegated zones. Zone keys must have the same name as the zone, a name type of **ZONE**, and must be usable for authentication. On z/OS, you should use the RSA algorithm for DNSSEC if the zone you will sign with the key will be a dynamic zone (that is, one maintained with nsupdate). You can also use the DSA algorithm, provided you have the 'random-device' option coded in the named.conf file.

The following command will generate a 768–bit RSA key for the child.example zone:

```
dnssec-keygen -a RSA -b 768 -n ZONE child.example
```

Two output files will be produced: Kchild.example.+001+12345.key and Kchild.example.+001+12345.private (where 12345 is an example of a key tag). The key file names contain the key name (child.example.), algorithm (3 is DSA, 1 is RSA, etc.), and the key tag (12345 in this case). The private key (in the .private file) is used to generate signatures, and the public key (in the .key file) is used for signature verification.

To generate another key with the same properties (but with a different key tag), repeat the above command.

The public keys should be inserted into the zone file with **$INCLUDE** statements, including the .key files.

Creating a key set

The **dnssec-makekeyset** program is used to create a key set from one or more keys.

Once the zone keys have been generated, a key set must be built for transmission to the administrator of the parent zone, so that the parent zone can sign the keys with its own zone key and correctly indicate the security status of this zone. When building a key set, the list of keys to be included and the TTL of the set must be specified, and the desired signature validity period of the parent's signature may also be specified.

The list of keys to be inserted into the key set may also include non-zone keys present at the top of the zone. dnssec-makekeyset may also be used at other names in the zone.

The following command generates a key set containing the above key and another key similarly generated, with a TTL of 3600 and a signature validity period of 10 days starting from now.

```
dnssec-makekeyset -t 3600 -e +8640 Kchild.example.+001+12345
Kchild.example.+001+23456
```

One output file is produced: keyset-child.example. This file should be transmitted to the parent to be signed. It includes the keys, as well as signatures over the key set generated by the zone keys themselves, which are used to prove ownership of the private keys and encode the desired validity period.

Signing the child's key set

The **dnssec-signkey** program is used to sign one child's key set.

If the child.example zone has any delegations which are secure, for example, grand.child.example, the child.example administrator should receive key set files for each secure subzone. These keys must be signed by this zone's zone keys.

The following command signs the child's key set with the zone keys:

```
dnssec-signkey keyset-grand.child.example. Kchild.example.+001+12345
Kchild.example.+001+23456
```

One output file is produced: signedkey-grand.child.example.. This file should be both transmitted back to the child and retained. It includes all keys (the child's keys) from the key set file and signatures generated by this zone's zone keys.

Signing the zone

The **dnssec-signzone** program is used to sign a zone.

Any **signedkey** files corresponding to secure subzones should be present, as well as a signedkey file for this zone generated by the parent (if there is one). The zone signer will generate NXT and SIG records for the zone, as well as incorporate the zone key signature from the parent and indicate the security status at all delegation points.

The following command signs the zone, assuming it is in a file called zone.child.example. By default, all zone keys which have an available private key are used to generate signatures.

```
dnssec-signzone -o child.example zone.child.example
```

One output file is produced: zone.child.example.signed. This file should be referenced by named.conf as the input file for the zone.

Configuring servers

Data is not verified on load in BIND 9, so zone keys for authoritative zones do not need to be specified in the configuration file. The public key for any security root must be present in the configuration file's **trusted-keys** statement.

IPv6 support in BIND 9

BIND 9 fully supports all currently defined forms of IPv6 name to address and address to name lookups.

For forward lookups, BIND 9 supports both A6 and AAAA records. The use of AAAA records is recommended, as A6 records might be moved to experimental status by RFC. In fact, the stub resolvers currently shipped with most operating systems support only AAAA lookups, because following A6 chains is much harder than doing A or AAAA lookups.

For IPv6 reverse lookups, BIND 9 supports the standard *nibble* label format, as well as the experimental *bitstring* format. Both formats are used under the ip6.arpa domain, although some resolvers and applications use the nibble format under the deprecated ip6.int domain.

Address lookups using AAAA records

The AAAA record is a parallel to the IPv4 A record. It specifies the entire address in a single record. For example:

```
$ORIGIN example.com.
host  3600 IN AAAA 3ffe:8050:201:1860:42::1
```

Address lookups using A6 records

A6 records are supported, but might be moved to experimental status by RFC. The use of AAAA records is recommended.

The A6 record is more flexible than the AAAA record, and is therefore more complicated. The A6 record can be used to form a chain of A6 records, each specifying part of the IPv6 address. It can also be used to specify the entire record as well. For example, this record supplies the same data as the AAAA record in the previous example:

```
$ORIGIN example.com.
host  3600 IN A6 0 3ffe:8050:201:1860:42::1
```

A6 chains: A6 records are designed to allow network renumbering. This works when an A6 record only specifies the part of the address space the domain owner controls. For example, a host may be at a company named ″company.″ It has two ISPs which provide IPv6 address space for it. These two ISPs fully specify the IPv6 prefix they supply.

In the company's address space:

```
$ORIGIN example.com.
host  3600 IN A6 64 0:0:0:0:42::1 company.example1.net.
host  3600 IN A6 64 0:0:0:0:42::1 company.example2.net.
```

ISP1 will use:

```
$ORIGIN example1.net.
company  3600 IN A6 0 3ffe:8050:201:1860::
```

ISP2 will use:

```
$ORIGIN example2.net.
company  3600 IN A6 0 1234:5678:90ab:fffa::
```

When host.example.com is looked up, the resolver (in the caching name server) will find two partial A6 records, and will use the additional name to find the remainder of the data.

Note: A6 chain resolution occurs when one name server requests an A6 record from another name server, with a query for a host name found in a previous A6 record. The name server can also perform A6 chain resolution on behalf of a resolver that cannot perform this function, including the z/OS resolver. To enable this function, the name server must be configured with the 'allow-v6–synthesis' option. Resolvers that cannot send A6 type queries and can only send AAAA type queries (like the z/OS resolver) are then able to follow A6 chains using a name server configured with 'allow-v6–synthesis'.

A6 records for DNS servers: When an A6 record specifies the address of a name server, it should use the full address rather than specifying a partial address. For example:

```
$ORIGIN example.com.
@            14400           IN      NS       ns0
             14400           IN      NS       ns1
ns0          14400           IN      A6       0 3ffe:8050:201:1860:42::1
ns1          14400           IN      A        192.168.42.1
```

It is recommended that IPv4-mapped IPv6 addresses not be used. If a host has an IPv4 address, use an A record, not an A6 record with an address like ::ffff:192.168.42.1.

Synthetic IPv6 responses

Most resolvers support IPv6 DNS lookups as defined in RFC 1886, using AAAA records for forward lookups and nibble labels in the ip6.int domain for reverse lookups, but do not support RFC 2874-style lookups (using A6 records and binary labels in the ip6.arpa domain). BIND 9 provides a way to automatically convert RFC 1886-style lookups into RFC 2874-style lookups and return the results as synthetic AAAA and PTR records.

This feature is disabled by default and can be enabled on a per-client basis by adding an allow-v6-synthesis *{ address_match_list }*; clause to the options or view statement. When it is enabled, recursive AAAA queries cause the server to first try an A6 lookup, and if that fails, an AAAA lookup. No matter which one succeeds, the results are returned as a set of synthetic AAAA records. Similarly, recursive PTR queries in ip6.int will cause a lookup in ip6.arpa using binary labels, and if that fails, another lookup in ip6.int. The results are returned as a synthetic PTR record in ip6.int. To enable allow-v6-synthesis for all clients, use the 'any' built-in ACL.

The synthetic records have a TTL of zero. DNSSEC validation of synthetic responses is not currently supported; therefore responses containing synthetic RRs will not have the AD flag set.

Address to name lookups using nibble format

When looking up an address in nibble format, the address components are simply reversed, just as in IPv4, and ip6.arpa. is appended to the resulting name. For example, the following would provide reverse name lookup for a host with address 3ffe:8050:201:1860:42::1.

```
$ORIGIN 0.6.8.1.1.0.2.0.0.5.0.8.e.f.f.3.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.2.4.0.0   14400 IN  PTR
host.example.com.
```

Address to name lookups using bitstring format

Bitstring labels can start and end on any bit boundary, rather than on a multiple of 4 bits as in the nibble format.

To replicate the previous example using bitstrings:

```
$ORIGIN \[x3ffe805002011860/64].ip6.arpa.
\[x0042000000000001/64]  14400  IN  PTR
host.example.com.
```

Using DNAME for delegation of IPv6 reverse addresses

DNAME is supported, but is considered experimental.

In IPv6, the same host may have many addresses from many network providers. Since the trailing portion of the address usually remains constant, **DNAME** can help reduce the number of zone files used for reverse mapping that need to be maintained.

For example, consider a host which has two providers (example.net and example2.net) and therefore two IPv6 addresses. Since the host chooses its own 64 bit host address portion, the provider address is the only part that changes:

```
$ORIGIN example.com.
host   A6 64 ::1234:5678:1212:5675
cust1.example.net.
   A6 64 ::1234:5678:1212:5675
subnet5.example2.net.
$ORIGIN example.net.
cust1   A6 48 0:0:0:dddd::
ipv6net.example.net.
ipv6net   A6 0 aa:bb:cccc::
$ORIGIN example2.net.
subnet5   A6 48 0:0:0:1::
ipv6net2.example2.net.
ipv6net2  A6 0 6666:5555:4::
```

This sets up forward lookups. To handle the reverse lookups, the provider example.net would have:

```
$ORIGIN \[x00aa00bbcccc/48].ip6.arpa.
\[xdddd/16]  DNAME  ipv6-rev.example.com.
```

and example2.net would have:

```
$ORIGIN \[x666655550004/48].ip6.arpa.
\[x0001/16]  DNAME  ipv6-rev.example.com.
```

example.com needs only one zone file to handle both of these reverse mappings:

```
$ORIGIN ipv6-rev.example.com.
\[x1234567812125675/64] PTR  host.example.com.
```

Advanced BIND 4.9.3–Name server topics

Connection optimization in a sysplex domain

This section describes *connection optimization*, a technique that uses DNS for balancing IP connections and workload in a sysplex domain. It also dynamically manages the active and available list of IP addresses for resources in a sysplex domain.

The name server can perform Connection Optimization (DNS/WLM) only when running in BIND 4.9.3 mode. Those wishing to do sysplex load balancing while running the name server in BIND 9 mode are encouraged to use the Sysplex Distributor function as an alternative.

You may run a BIND 9 and a BIND 4.9.3 name server simultaneously; however, the host's interfaces must be divided among the two. That is, clients wishing to be served by a BIND 4.9.3 name server must send queries to the BIND 4.9.3 addresses and clients wishing to be served by a BIND 9 name server must send queries to the BIND 9 addresses. If you wish to run BIND 4.9.3 and BIND 9 name servers simultaneously while using DNS/WLM (Connection Optimization) with the BIND 4.9.3 name server, you can still make all addresses on the host available to the BIND 4.9.3 name server for Connection Optimization. You do not have to limit that set of addresses to the set of addresses that the BIND 4.9.3 name server is listening on. The mutually exclusive set of addresses that the two name servers are listening on is independent of the addresses that the name servers may return to clients. Refer to *z/OS Communications Server: IP Migration* for more information.

In BIND 9 mode, primitive load balancing without Sysplex Distributor can be achieved in DNS using multiple A records for one name, if true sysplex load balancing is not desired through the Sysplex Distributor. In this case, the addresses from the multiple A records will be handed out in a round-robin fashion.

Overview

Connection optimization uses DNS for distributing connections among hosts or server applications within a sysplex domain. A sysplex is a set of MVS systems communicating and cooperating with each other through multisystem hardware and software components.

In DNS terms, a sysplex is a subdomain that is added to the DNS name space. Name servers running within the sysplex perform name resolution. Resolvers query these name servers directly or indirectly through the name server authoritative for the resources in the sysplex domain.

Connection optimization extends the concept of a ″DNS host name″ to clusters, or groups of server applications or hosts. Server applications within the same group are considered to provide equivalent service. Connection optimization utilizes round-robin logic and load-based ordering to determine which addresses to return for a given cluster.

Connection optimization increases overall efficiency by favoring connections to systems with the most available resources and by avoiding unavailable sysplex resources. Addresses of the most available server applications or hosts are returned more frequently than the addresses of loaded server applications or hosts.

A connection-optimized sysplex domain is also scalable—that is, it can add servers and interface addresses dynamically to provide more service capacity. Client applications have dynamic access to the addresses of those servers, with no DNS restart or administration required.

Registration: To ensure maximum availability, server applications register with Workload Manager (WLM), which quantifies the availability of server resources within a sysplex. WLM must be configured in goal mode on all hosts within the sysplex. See "Step 7: Configure WLM in goal mode" on page 495 for a description of this procedure.

TCP/IP stacks also register with WLM. Additionally, they provide the active IP addresses. For a description of how IP addresses are associated with a sysplex domain name, see "Associating IP addresses with the sysplex domain name" on page 485. For a discussion of TCP/IP configuration issues, see "Configuring TCP/IP" on page 491.

When registering, server applications provide the following information:

- *Group name*. This is the name of a cluster of equivalent server applications in a sysplex. It is also the name within the sysplex domain that client applications use to access the server applications. To connect to *any* server application in a group, a client application uses the combination *group_name.sysplex_domain_name*.

  Note: The group name *TCPIP*, the group name for the sysplex domain (for example, *mvsplex*), and the DNS names of the resource records in your sysplex *cluster* zone file are reserved and cannot be used by server applications.
- *Server name*. This is the name of the server application instance. The server name must be unique among all servers that share the same group name. A server application instance can belong to more than one group.
- *Host name*. This is the host name of the TCP/IP stack on which the server application runs.

The sysplex domain name should be registered with the domain name server under a special address, 127.0.0.128, which is used by the ioctl() SIOCGSPLXFQDN. For details, see "Configuring a master (primary) name server" on page 427.

Name resolution: In connection optimization, a name server performs resolution for a name representing a cluster of hosts or server applications. Figure 58 on page 484 depicts a sysplex domain called `mvsplex.mycorp.com`. The sysplex domain contains only the resources participating in the sysplex. Client applications append the domain name, `mycorp.com`, to all requests for name resolution.

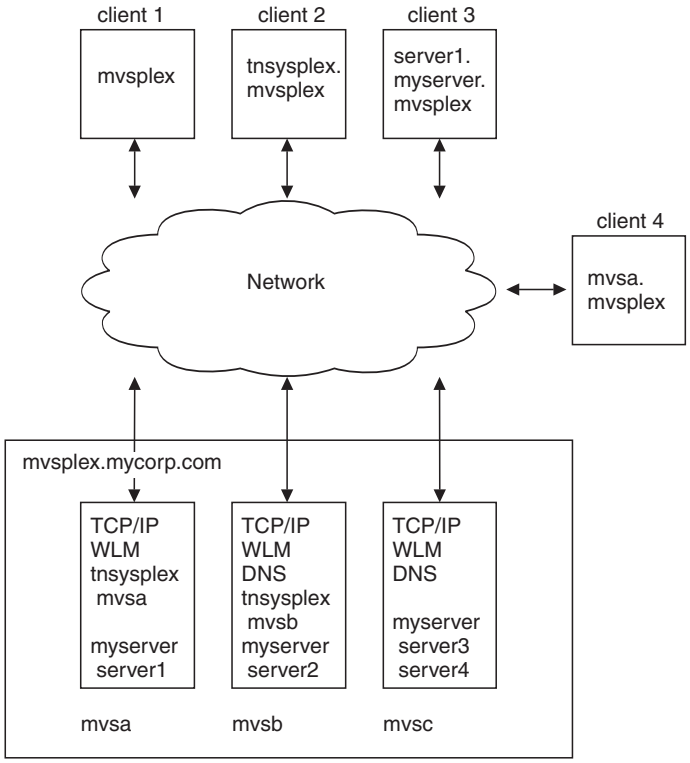


Figure 58. Name resolution to a sysplex

Each host system runs TCP/IP. Hosts `mvsb` and `mvsc` are also running the z/OS UNIX sysplex name servers (`DNS`). `mvsb` is running the master name server for the sysplex subdomain, and `mvsc` is running the slave.

Each host is running one or more `myserver` server applications, and `mvsa` and `mvsb` are also running `tnsysplex` server applications. The group names (`tnsysplex` and `myserver`), the `tnsysplex` server names (`mvsa` and `mvsb`), and the `myserver` server names (for example, `server1`) are known to the name server through WLM registration by the respective application.

Note: `tnsysplex` represents a cluster of TN3270 server applications. Every instance of TN3270 running on a particular host registers using the same server name.

Four types of requests are shown in Figure 58. Client Application 1 requests the services of any host on the system by providing only the name of the sysplex subdomain, `mvsplex`, to DNS for resolution. Client Application 2 requests the services of any `tnsysplex` server instance running in the sysplex. Client Application 3 requests a particular server instance in the `myserver` group, and Client Application 4 requests connection to a particular host.

In Figure 58, only Client Applications 1 and 2 are candidates for connection optimization. Client Application 1 can connect to either `mvsa`, `mvsb`, or `mvsc`. Client Application 2 can connect to either `mvsa` or `mvsb` (but not `mvsc`). Client Applications 3 and 4, which can connect only to `mvsa`, are ineligible for connection optimization. They do, however, benefit from the round-robin selection process DNS uses to balance across available network interfaces.

Note: When using connection balancing via MVS clients (such as FTP), to avoid caching of IP addresses, during name resolution set the following environment variable:

```
export _EDC_IP_CACHE_ENTRIES=0
```

Generated names vs. statically defined names: All name servers use *statically defined* names. These are the names in the forward domain data file. As the DNS administrator for a name server using connection optimization, statically define the names of the hosts in the sysplex and the NS and SOA resource records in the forward domain data file. For a description of sysplex forward domain data files, see "Sysplex data files" on page 493.

Note: The host names in the data files must match the host names specified in the stack's TCPIP.DATA data set and should be 20 characters or less to ensure server uniqueness.

A name server using connection optimization also uses *generated names*. These are added dynamically to the domain name space as TCP/IP stacks and server applications in the sysplex register with WLM. The name server uses three types of generated names. In Figure 58 on page 484, the following generated names are used:

- The group name that is registered by the server applications (`tnsysplex` and `myserver`)
- The server name concatenated with the group name of each server application that registers with WLM in the sysplex (for example, `server1.myserver`)
- An alias for the sysplex domain name, `mvsplex`

The generated names become resources (or host names) within the sysplex domain, creating fully qualified domain names:

- Fully qualified group names (these are the connection balanced names):
  - `tnsysplex.mvsplex.mycorp.com`
  - `myserver.mvsplex.mycorp.com`
- Fully qualified server names:
  - `mvsa.tnsysplex.mvsplex.mycorp.com`
  - `server1.myserver.mvsplex.mycorp.com`
- Fully qualified alias name for the sysplex name `mvsplex.mycorp.com`:
  - `mvsplex.mvsplex.mycorp.com`

Associating IP addresses with the sysplex domain name: The sysplex domain name `mvsplex.mycorp.com` is associated with the *intersection* of the set of statically defined addresses associated with the stacks that are registered with WLM and the set of addresses associated with the adapters that are active on those stacks. The TCP/IP stack must be registered with WLM for this to occur. Figure 59 on page 486 depicts this intersection.

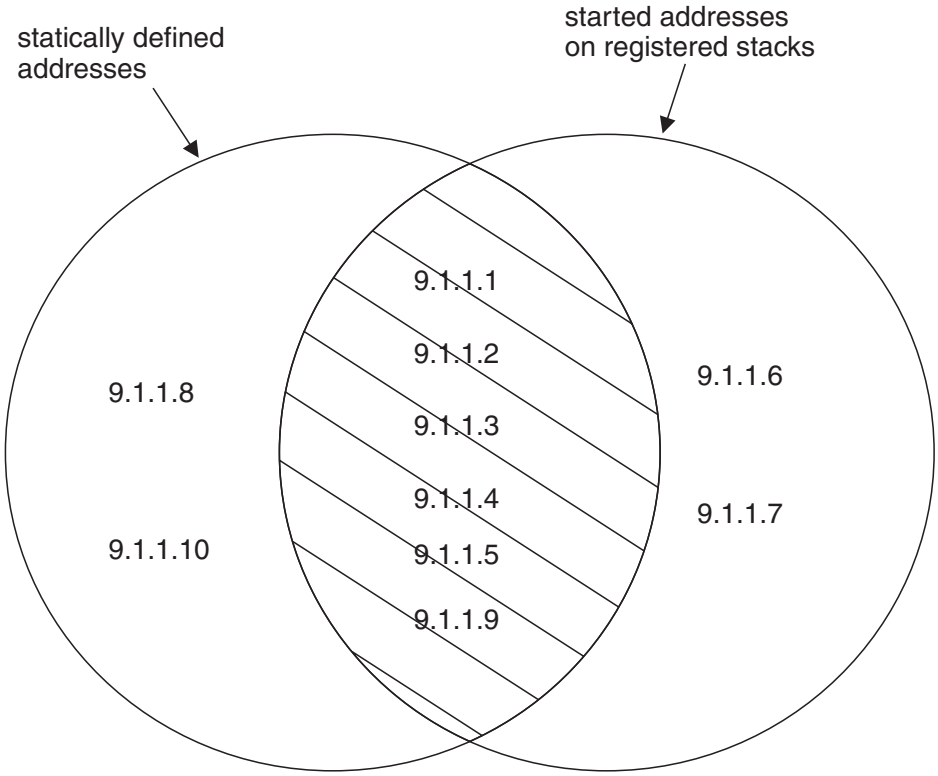


Figure 59. Address association with mvsplex.mycorp.com

The hosts, `mvsa`, `mvsb`, and `mvsc` in the figure have the following addresses in the HOME statement in the PROFILE.TCPIP data set. Only certain adapters are active.

```
mvsa                    mvsb                      mvsc
9.1.1.1 (adapter active) 9.1.1.4 (adapter active)   9.1.1.8
9.1.1.2 (adapter active) 9.1.1.5 (adapter active)   9.1.1.9 (adapter active)
9.1.1.3 (adapter active) 9.1.1.6 (adapter active)   9.1.1.10
                         9.1.1.7 (adapter active)
```

The forward domain data file for the sysplex contains the following statically defined addresses:

```
mvsa            IN        A         9.1.1.1
                IN        A         9.1.1.2
                IN        A         9.1.1.3
mvsb            IN        A         9.1.1.4
                IN        A         9.1.1.5
mvsc            IN        A         9.1.1.8
                IN        A         9.1.1.9
                IN        A         9.1.1.10
```

Using the intersection of two sets lets the domain administrator selectively exclude certain IP addresses from use, such as 9.1.1.6 and 9.1.1.7 on mvsb, while distributing only active addresses to client applications. For instance, 9.1.1.8 would never be used since it is not active.

The process of associating IP addresses with server applications is similar to that for hosts. When a server application registers with WLM, the dynamically generated group name (for example, `myserver`) is added to the sysplex domain. If the stack on which the server application is running is registered with WLM, then the addresses associated with the group name are the intersection of the statically defined addresses for that stack and those addresses that are associated with adapters that

are active. When the server application is replicated on other hosts in the sysplex, the addresses associated with the group name are the union of all intersection sets.

Detailed example: The following example describes in detail how IP addresses become associated with a server application `myserver` running in the sysplex `mvsplex.mycorp.com`. When reading the following section, refer to Figure 60.

The example is described in terms of a *process*, beginning initially with no registered servers applications or stacks. The statically defined addresses associated with the `mvs` hosts and coded in the forward domain data file are the statically defined addresses listed earlier. As server applications and stacks register, the IP addresses associated with `myserver.mvsplex.mycorp.com` change. Figure 60 shows the conclusion of this process.

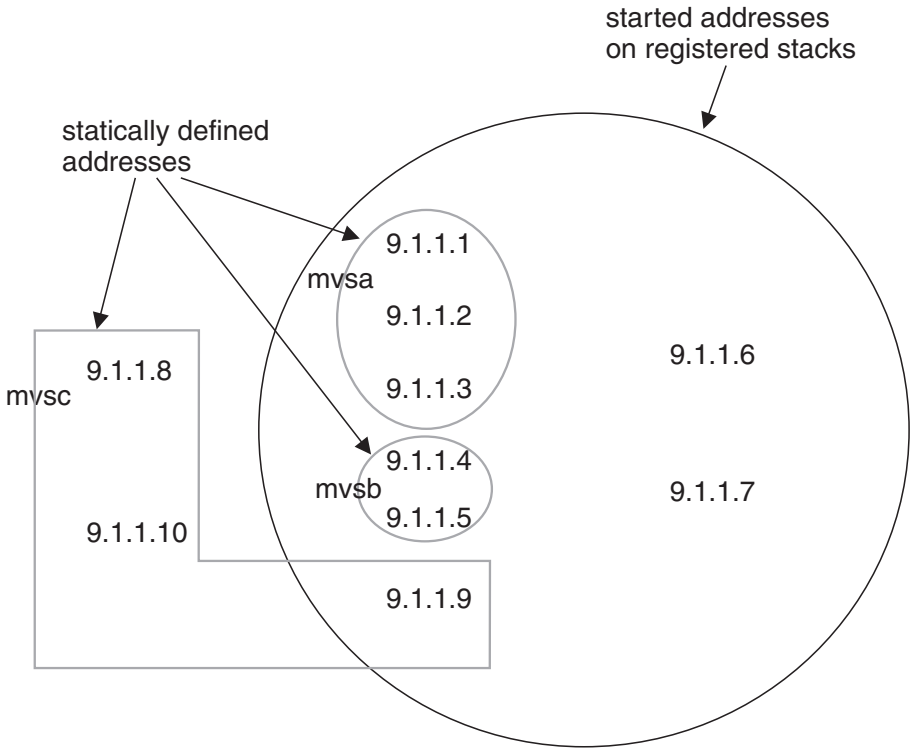


Figure 60. Address association with myserver

Assume initially that a server application which registers with the group name `myserver` is running on `mvsc`, but that the stack on `mvsc` is not currently registered with WLM. The addresses associated with the name `myserver.mvsplex.mycorp.com` are `9.1.1.8`, `9.1.1.9`, and `9.1.1.10`.

Next, assume that the stack on `mvsc` registers with WLM. The addresses associated with the name `myserver.mvsplex.mycorp.com` are reduced to the intersection of the statically defined addresses for the stack and those addresses that are associated with adapters that are active. The only address associated with group name `myserver.mvsplex.mycorp.com` is thus `9.1.1.9`.

Now another instance of an equivalent server application registers with WLM on `mvsb`, but the stack on `mvsb` is *not* registered with WLM. All the statically defined addresses associated with `mvsb` are added to the set of addresses currently

associated with the group name myserver.mvsplex.mycorp.com (9.1.1.4, 9.1.1.5, and 9.1.1.9). If the addresses on mvsb are active, the set of addresses associated with the group name does not change.

Similarly, if another instance of an equivalent server application registers with WLM on mvsa and if the addresses on mvsa are active, the set of addresses associated with the server application myserver.mvsplex.mycorp.com are those shown in Figure 60 on page 487 (and Figure 59 on page 486).

Note: The adapters associated with addresses 9.1.1.6 and 9.1.1.7 are not associated with the dynamically generated group name because they are not statically defined in the forward domain data file.

Connecting to a particular server instance: The server name with which the server application registers is unique among all other server instances that share the same group name. Client applications can use the server name to bypass connection optimization. For example, client applications can bypass connection optimization if they are in the middle of a transaction with a server application, and the client/server session fails before the transaction completes. If the client application software has the ability to recognize this situation, the client can reconnect to the same server instance and complete the transaction.

The group name is prefaced with the server name, separated by a dot (server_name.group_name). If the group name is myserver and if the application instance on mvsc registered with WLM as myserver3, then client applications can connect to that particular instance using the name myserver3.myserver.mvsplex.mycorp.com. The address associated with this name (9.1.1.9) is a subset of the addresses currently associated with the group name myserver.mvsplex.mycorp.com that exist on stack mvsc.

Usage considerations in a connection optimized sysplex: Connection optimization extends the concept of a DNS host name to include a name that generically represents (1) all hosts in the sysplex (provided their stacks register with WLM) and (2) names that represent groups of equivalent server applications spread across the sysplex (provided those server applications register with WLM). The maximum benefit from connection optimization is realized when all stacks in the sysplex register with WLM and all TCP/IP server applications register with WLM, if they are capable of doing so.

To take advantage of connection optimization even when registration is not available, consider the following scenarios.

TCP/IP server application does not register: In some cases, you might want to use connection optimization for a particular TCP/IP server application, but it does not register with WLM. If the *stacks* that these applications are running on register with WLM, users can still use connection optimization with server applications by entering the sysplex domain name. A typical invocation might be the following:

```
tn3270 mvsplex
```

where tn3270 is the name of the invoked application and mvsplex is the sysplex domain name.

In this scenario, system administrators must ensure that equivalent instances of the server application are running on each registered stack. Otherwise, connection optimization might result in connecting the client application to a host that is not

running the server application. (mvsc in Figure 58 on page 484, for example, is not running the TN3270 server.) Depending on the client software, connection timeouts and connection retries might result.

One or more (or all) stacks do not register with WLM: In some cases, you might want to use connection optimization for a particular TCP/IP server application, but one or more (or all) stacks do not register with WLM because they do not support WLM registration or they are not configured to do so. Consider also that the server application does not register with WLM. For the stacks that have not registered, only the statically defined addresses will be used.

In this scenario, a certain degree of connection optimization can occur between hosts whose stacks register with WLM if users enter the sysplex domain name (for example, mvsplex in Figure 58 on page 484.) In addition, system administrators must ensure that equivalent instances of the server application are running on each registered stack. Connections will not be made to hosts whose stacks do not register.

Similarly, suppose that the application programmer develops a server application called ourApp that registers with the group name of myserver. Suppose also that one or more (or all) of the stacks on which the server application runs are not registered with WLM. A typical invocation might be the following:

```
ourApp myserver
```

Since one or more (or all) of the stacks are not registered with WLM, it is possible that an unusable IP address could be returned to the client because a stack has not reported the IP addresses that are active. In this case, only statically defined addresses are used for stacks that are not registered. If an unusable IP address is returned to the client, the connection times out, and depending on the client software, the client application might or might not retry the same or different address returned by the DNS query.

Considerations for connection balancing with multiple instances of TCP/IP on a single host system: Results of connection balancing in a multiple TCP/IP environment are not predictable. Servers such as TN3270 that bind to a single instance of TCP/IP should work as expected as long as they provide the correct host name in the WLM registration. Servers such as FTP which do not bind to a single instance of TCP/IP will have less predictable results. The host name provided on the registration will link each server with a set of IP addresses for a single stack.

Multiple servers on the same port: When running multiple servers on the same port on the same TCP/IP instance (using SHAREPORT), only one of the servers should register with WLM.

Caching issues: Proper distribution of server application addresses within a cluster requires DNS queries to be answered by the name server within the sysplex. For this reason, name servers that are located outside the sysplex cannot be configured as master or slave servers for the sysplex domain.

Name servers or resolvers outside the sysplex can prevent client application queries from reaching the sysplex name servers on subsequent requests if they use cached information. This is undesirable for connection optimization since the name servers and resolvers would not have up-to-date information about capacity and availability of the resources in the sysplex.

To disable other name servers from caching information about sysplex domain resources, a time-to-live (ttl) value of 0 is returned by default. Note that some resolver and name server implementations do not support a ttl value of 0 or anything less than an internally defined minimum (for example, 300 seconds).

Depending on the DNS and network configuration, the number of DNS queries on the network for the sysplex resources might increase. At the expense of reduced availability and load distribution information, administrators can choose a different default ttl for the sysplex resources by using the `-l` option when starting the sysplex name server.

Configuring a sysplex domain for connection optimization

Follow the steps below to configure name servers in a sysplex domain:

1. Identify server applications.
2. Configure server applications for WLM registration.
3. Choose sysplex name and identify name servers in the sysplex.
4. Update parent domain name server.
5. Configure the sysplex name servers.
6. Configure client applications.
7. Configure WLM in goal mode.

Each of these steps is explained below.

Step 1: Identify server applications: Identify the server applications to run in the sysplex. Refer to the product documentation for each application to determine if it supports registration with WLM for connection optimization. It is possible to modify the application to register with WLM. See "Registering your own applications" on page 495.

Candidate applications must have the following attributes:

- Client applications must use DNS for name resolution.
- Server applications must run in a single sysplex.
- Server applications within a specified group provide equivalent functions to their clients. That is, the client application receives the same services from any of the registered server applications.
- To be considered equivalent, all servers registering in a group must be listening on the same port.
- Client applications must use portmapper or a well-known port number to access the server application.

Note: Data Facility Storage Management Subsystem (DFSMS) restrictions do not currently allow sharing of HFS files in write mode.

Maximum benefits are attained when server applications have the following attributes:

- *Registration with WLM.* This feature allows WLM to track the availability of the registering server application and to allow clusters of servers on specified systems within the sysplex. Client application use of the sysplex domain name assumes that the server application is available on all hosts within the sysplex and the stacks running on the sysplex hosts are configured to register with WLM. Even if a server application does not register with WLM, it might still be able to take advantage of connection optimization under certain circumstances. See "Usage considerations in a connection optimized sysplex" on page 488.

- *Workloads.* The server application has system or network workloads sufficient enough to warrant load distribution. Determination of what is sufficient is subjective, but the value of balancing the incoming connections should outweigh the cost of the extra DNS queries on the network. See “Caching issues” on page 489 for more information.
- *Session Length.* In most cases, server applications selected to use the connection optimization model should have long sessions. Because reduced caching in the network name servers causes additional network traffic, server applications with many short sessions might not be appropriate candidates.

Step 2: Configure server applications for WLM registration: After identifying server applications you want to run in the sysplex, you configure them for WLM registration. Typical configuration involves specification of the group name by which the application will be known. (Do not use ″TCPIP″ or the sysplex domain name as a group name.) See “Registering your own applications” on page 495 for information about how to register the applications you write. The following sections describe how to configure the z/OS CS TCP/IP stack and the applications that are able to register with WLM.

Configuring TN3270: To configure TN3270 servers for registration with WLM, use the WLMCLUSTERNAME and ENDWLMCLUSTERNAME statements to enter a unique sysplex server group name (or names) in the TELNETPARMS section of the PROFILE.TCPIP data set. For a complete description of these statements, refer to *z/OS Communications Server: IP Configuration Reference*.

Unique TN3270 server group names can be used to separate or direct client application requests to only those host systems supporting the target application. For example, a TN3270 server group name of `CICS3270` could be used on host systems actually running the target CICS application that the TN3270 clients access.

If necessary, modify the server group names using the VARY TCPIP,,OBEYFILE command. (The VARY TCPIP,,OBEYFILE command allows changes to the system operation and network configuration without stopping and restarting the TCP/IP address space.) If using VARY TCPIP,,OBEYFILE, however, specify *all* of the TN3270 parameters between the TELNETPARMS and ENDTELNETPARMS statements (not just additions and deletions). For more information on VARY TCPIP,,OBEYFILE processing, refer to *z/OS Communications Server: IP Configuration Reference*.

Configuring TCP/IP: To register TCP/IP with WLM at startup, add the IPCONFIG SYSPLEXROUTING statement in the PROFILE.TCPIP configuration data set. To register TCP/IP using VARY TCPIP,,OBEYFILE, add the IPCONFIG SYSPLEXROUTING statement to the OBEYFILE data set. For a complete description of this statement, refer to *z/OS Communications Server: IP Configuration Reference*.

TCP/IP also registers addresses that are active for each stack. These addresses are the active, configured addresses in the `HOME` list for which a `START` device has been processed. WLM registration supports only 15 addresses per TCPIP instance.

To deregister TCP/IP, add IPCONFIG NOSYSPLEXROUTING to the OBEYFILE data set. (Although it is possible to add this statement to the PROFILE.TCPIP data set, typical de-registration occurs in the OBEYFILE data set.) When using VARY TCPIP,,OBEYFILE processing with IPCONFIG NOSYSPLEXROUTING, add

IPCONFIG SYSPLEXROUTING on a subsequent VARY TCPIP,,OBEYFILE command to reregister. Note that de-registration occurs automatically when TCP/IP terminates.

Note: Use the IPCONFIG SYSPLEXROUTING and IPCONFIG NOSYSPLEXROUTING statements only when the host is running as part of a sysplex.

Configuring the FTP server: Refer to *z/OS Communications Server: IP Configuration Reference* for more information on WLMCLUSTERNAME statement.

Configuring CICS: See *z/OS Communications Server: IP CICS Sockets Guide*.

Step 3: Choose sysplex name and identify name servers: Identify a unique name with which DNS client applications can access the sysplex. This name can be the same as the configured sysplex name. Names must be 18 characters or less to accommodate WLM restrictions. The sysplex name becomes part of the fully qualified domain name. For example, a sysplex, `mvsplex`, in the domain `mycorp.com` has a fully qualified name of `mvsplex.mycorp.com`. (This is the domain name you specify in the primary directive that contains the `cluster` keyword in the named boot file.)

After selecting a name for the sysplex, identify the master name servers. A sysplex should have only one master name server, and it should have a slave name server to provide redundancy. The slave name server must run in the same sysplex as the master name server. Both the master and slave name servers configured for connection optimization must run on hosts within the sysplex.

Step 4: Update parent domain name server: The parent domain name server is the master name server authoritative for the domain that contains the sysplex subdomain. The parent domain name server can be the same name server as the sysplex name server.

Update the parent domain name server data files by entering the names and IP addresses of the name servers for the sysplex. Use NS (Name Server) resource records to enter the information.

For example, if the name of the domain in which the sysplex is located is `mycorp.com` and the name of the sysplex is `mvsplex` and the master name servers are running on hosts `mvsb` and `mvsc` in the sysplex, add the following records to the forward domain data file of the master domain name server for `mycorp.com`:

```
mvsplex NS mvsb.mvsplex.mycorp.com.
        NS mvsc.mvsplex.mycorp.com.
mvsb.mvsplex.mycorp.com. A  9.67.116.201
                         A  9.67.116.206
                         A  9.67.116.208
mvsc.mvsplex.mycorp.com. A  9.67.116.203
                         A  9.67.116.207
                         A  9.67.116.210
```

This tells the parent domain name server that mvsb and mvsc are master servers for mvsplex. It does not tell the name server that mvsc is slave, only that it is an additional master name server in the sysplex. The address (A) records are glue records. They enable remote name servers to contact the name servers in the sysplex.

In addition, it might be useful to add CNAME records for resources within the sysplex subdomain to avoid a name change in the client application and to prevent confusion for the end user. See “Step 6: Configure client applications” on page 494.

For example, if client applications use a default domain name of mycorp.com, then requests for `tnsysplex.mycorp.com` can be delegated to the sysplex subdomain with the following resource record:

```
tnsysplex   CNAME   tnsysplex.mvsplex.mycorp.com.
```

Step 5: Configure the sysplex name servers: When the TCP/IP stack is registered with WLM, the list of IP addresses made available to client applications includes the addresses that are common to those provided by TCP/IP and the list of application servers in the forward domain data file of the sysplex name server. See “Generated names vs. statically defined names” on page 485.

Note: The addresses returned to the client application depend upon several factors including whether all or some stacks in the sysplex are registered with WLM, the availability of the adapters assigned to the IP addresses, the names the client application uses for connection, and whether the servers are registered with WLM. See “Usage considerations in a connection optimized sysplex” on page 488.

The steps for configuring sysplex name servers are similar to those for configuring the name servers in an ordinary subdomain:

“Step 2. For BIND 4.9.3–DNS only: specify stack affinity (Multiple stack environment)” on page 431.

“Step 3. Specify port ownership” on page 431.

“Step 4. Update the name server start procedure (Optional)” on page 432.

“Step 5. Create the domain data files (master name server only)” on page 433. See examples below for more information.

“Step 6. Create the hints (root server) file” on page 439.

“Step 7. Create the loopback file” on page 441.

“Step 1a. Create the boot file for BIND 4.9.3–DNS” on page 428.

“Step 11. Start the name server” on page 448.

Sysplex data files: The data files must contain the ″A″ resource records for the host names internal to the sysplex. The host names must match the host names specified in the stack’s TCPIP.DATA data set. For a discussion of how IP addresses are associated with a sysplex domain name, see “Associating IP addresses with the sysplex domain name” on page 485.

Note: Consider using VIPA addresses for the MVS hosts. If VIPA addresses are used, they are the only addresses needed to code in the forward domain data file. When Dynamic VIPAs (DVIPAs) are used for VIPA Takeover, code all of the DVIPAs in the sysplex under each host name in the forward domain data file. This will circumvent manual intervention in the DNS data files when a DVIPA is taken over or given back and will not cause any unexpected results in DNS/WLM.

Following is an example of a forward domain data file for a sysplex name server.

```
mvsplex.mycorp.com.  SOA mvsb.mvsplex.mycorp.com.
                              administrator@us.mycorp.com.  (
   1997061300  ; serial
   10800       ; refresh after 3 hours
```

```
   1800        ; retry 1/2 hour after failed zone transfer
   3600000     ; expire; length of time secondary keeps data unless refreshed
   259200 )    ; default TTL for static resource records = 3 days
;
;  Define nameservers
          IN  NS  mvsb.mvsplex.mycorp.com.
          IN  NS  mvsc.mvsplex.mycorp.com.
;
;  Define localhost
;
localhost IN  A   127.0.0.0
;
;  Hostnames specified here must match hostnames
;  specified in the stack's TCPIP.DATA file.
mvsa      IN  A     9.1.1.1
          IN  A     9.1.1.2
          IN  A     9.1.1.3
              TXT   "Text record"
              HINFO "3090" "OS/390R4"
mvsb      IN  A     9.1.1.4
          IN  A     9.1.1.5
          IN  A     9.1.1.6
          IN  A     9.1.1.7
mvsc      IN  A     9.1.1.8
          IN  A     9.1.1.9
          IN  A     9.1.1.10
```

Sysplex boot files: The boot file is the main configuration file for a name server. It points the name server to the domain data files, the loopback file, and the hints (root server) file. The contents and format of boot files for sysplex name servers are identical to those for the boot files of ordinary master servers with the exception of an additional keyword, `cluster`. This keyword is used only once in a boot file, at the end of either the primary or secondary directive to identify the sysplex domain.

The following is a sample boot file for the master name server in a sysplex:

```
directory /etc/dnsdata
primary    mvsplex.mycorp.com         named.wlm.for    cluster
primary    113.67.9.in-addr.arpa      named.rev
primary    0.0.127.in-addr.arpa        named.lbk
cache      .                          named.ca
```

The file `named.wlm.for` identifies the forward domain data file for the master name server in the sysplex.

The following is an example of a boot file for a slave name server in a sysplex:

```
directory   /u/usr35/plex/secondary/zone1
secondary   mvsplex.tcp.raleigh.ibm.com   9.67.116.200 mvsplex.bak cluster
secondary   116.67.9.in-addr.arpa         9.67.116.200 mvsplex.rev
primary     0.0.127.in-addr.arpa           db.127.0.0
cache       .                            named.ca
```

Step 6: Configure client applications: Since the sysplex domain is created as a subdomain of an existing domain, resolvers should be reconfigured if it is expected that the clients will use names in the sysplex domain. Otherwise, the client will be forced to use fully qualified domain names to resolve sysplex domain names. For example, if the resolver's default domain was `mycorp.com` before adding the sysplex domain, consider changing the resolver's default domain to `mvsplex.mycorp.com`. Also consider adding the sysplex domain to the resolver's search list if the client's resolver supports it.

See "Name resolution" on page 483 for the various names a client can specify.

Step 7: Configure WLM in goal mode: All hosts in a sysplex must operate in goal mode for proper load balancing (splitting). If hosts are not in goal mode, they are treated with equal weight and round-robin selection is applied.

Configure WLM in goal mode on a host by issuing the following MVS command:

```
F WLM,MODE=GOAL
```

Alternatively, IPL in goal mode by omitting the `IPS=` keyword from your IEASYSxx parmlib member and from your IEASYS00 parmlib member. See *z/OS MVS Programming: Workload Management Services*.

Registering your own applications

Register a server application with WLM using a C interface or an assembler interface. See "Step 1: Identify server applications" on page 490. The C function is invoked as follows:

```
extern long IWMDNREG( char *group_name,
                      char *host_name,
                      char *server_name,
                      char *netid,
                      char *wlm_user_data,
                      long *diag_code);
```

A sample header file, `iwmwdnsh.h`, comes with the product. Refer to the program directory for its location. The following definitions apply:

- *group_name* is the name client applications use. It can be up to 18 characters.
- *host_name* is the TCP/IP name of the host on which the server is running. See *z/OS C/C++ Programming Guide*.
- *server_name* is a unique name that defines a particular instance of the server. It can be up to eight characters.
- *netid* and *wlm_user_data* should be null pointers.

Return values and diag_code values are documented in *z/OS MVS Programming: Workload Management Services* .

To register with a macro, use the IWMSRSRG macro. For a description of this macro, see *z/OS MVS Programming: Workload Management Services*. When using this macro, note the following:

- *Location* contains the *group_name*

  Note: The group name *TCPIP*, the group name for the sysplex domain (for example, *mvsplex*), and the DNS names of the resource records in your sysplex *cluster* zone file are reserved and cannot be used by server applications. If this rule is violated, you will receive message EZZ6649E:

  ```
  WLM group group_name not created
  ```

- *Network_ID* can be blank.
- *LUName* contains the *server_name*.
- *Host* contains the TCP/IP host name.

You can deregister a server application from WLM using a C interface or an assembler interface. Deregister whenever you do not want the server to receive additional client application connections. The C function is invoked as follows:

```
extern long IWMDNDRG( char *group_name,
                      char *host_name,
                      char *server_name,
                      char *netid,
                      long *diag_code);
```

To deregister with a macro, use the IWMSRDRS macro. For a description of this macro, see *z/OS MVS Programming: Workload Management Services*.

For C interfaces for WLM registration and de-registration calls, see the sample registration file, /usr/lpp/tcpip/samples/wlmreg.c.

Dynamic IP

DHCP on z/OS and OS/2 is only compatible with BIND 4.9.3. This section describes the purpose of Dynamic IP and its benefits. Also included is an introduction to the Dynamic IP components and an overview of design concepts.

DHCP on other platforms and Windows 2000 Active Directory can be compatible with BIND 9.

Overview

To add a new workstation to an IP network, several parameters and a variety of information is required to configure the TCP/IP software. Network components, such as a domain name server, are also required. A new TCP/IP host would normally require the following information:

- IP address
- IP subnet mask
- Default router address
- Local host name
- Domain name
- Name server address

Additional parameters, such as other server addresses, time zones, or protocol-specific configurations, might also be necessary.

Keeping track of that information in a large TCP/IP network is not an easy task for network administrators, especially if users or machines or both change their location frequently. IP address lists and domain name server databases have to be updated manually to keep track of any changes in the network.

From a user's point of view, a system administrator would have to be called to provide the necessary information to install a TCP/IP system. If the user moves to another location, this information must not be taken; the user will have to be assigned at least a new IP address if not a new host name as well. Thus, users could potentially cause disorder in a TCP/IP network.

Even if workstations will be automatically installed using software distribution techniques, the TCP/IP configuration parameters have to be pre-assigned per distribution client.

The Bootstrap Protocol (BootP), as described in RFCs 951 and 1497, was introduced to the TCP/IP community in 1985 to provide automatic assignment of some TCP/IP configuration parameters to a new TCP/IP host. A table has to be maintained at BootP servers to enter information specific to any client that has been planned for installation. Typically, clients are identified by their LAN adapter's

hardware address, which has to be known to the system administrator in charge of a BootP server before preparing a new client entry in the database. Even though some manufacturers put the adapter hardware address on a label on the backplane of their LAN adapters, this is a tedious process if many hosts have to be installed in a short period of time.

Objectives and customer benefits of dynamic IP: To lessen the problems of having to manually update any centrally maintained information files and of having a user manually configure a TCP/IP workstation, the Dynamic Host Configuration Protocol (DHCP) has been designed and is described in RFCs 1533, 1534, 1541, and 1542. A DHCP server need not be preconfigured with a workstation's LAN address to submit the necessary TCP/IP configuration to it.

With DHCP in place, the assignment of IP addresses is a lot easier, but one problem persists—how would a domain name server learn about dynamically assigned IP addresses and host names so it can update its database accordingly? This can be solved by the Dynamic Domain Name Services (DDNS).

IBM is actively participating in the designs and implementations of DHCP and DDNS, and it has coined the term *Dynamic IP*. To summarize, the objectives of Dynamic IP and its benefits to TCP/IP system administrators and users are as follows:

- Provides automatic IP network access and host configuration
- Simplifies IP network administration
- Leverages existing IP network products and infrastructure
- Employs only open standards
- Allows customers to administer site-specific host environments
- Enables customized, location-sensitive parameter setups

Note: For further information on the Dynamic Host Configuration Protocol (DHCP) server, see "Configuring the DHCP server for z/OS" on page 501.

Dynamic IP components: Table 20 gives a brief description of the four types of network components that comprise Dynamic IP.

Table 20. DHCP server configuration

System components	Description
Dynamic IP Hosts	Dynamic IP hosts contain DHCP client software and might contain Dynamic DNS client software. Together, they discover and cooperate with their DHCP and Dynamic DNS server counterparts in the network to automatically configure the hosts for network participation.
DHCP Servers	DHCP servers provide the addresses and configuration information to DHCP and BootP clients on the network. DHCP servers contain information about the network configuration and about host operational parameters, as specified by the network administrator. DHCP server can also be configured to be the proxy for the DDNS client and issue the commands to update the Dynamic DNS server.
DDNS Servers	Dynamic DNS servers are a superset of static DNS servers. The dynamic enhancements enable client hosts to dynamically register their name and address mappings in the DNS tables directly, rather than having an administrator manually perform the updates.

Table 20. DHCP server configuration (continued)

System components	Description
BootP Relay Agents (or BootP Helpers)	BootP relay agents can be used in IP router products to pass information between DHCP clients and servers. BootP relays eliminate the need for having a DHCP server on each subnet to service broadcast requests from DHCP clients.

Administering dynamic domains

The DDNS server performs Dynamic DNS database updates in the appropriate domain file as the updates occur. Therefore, **do not edit domain files for dynamic zones while the DDNS server is running.** Furthermore, dynamic domains cannot be dynamically reinitialized with new configuration information using the traditional **nssig -HUP** command.

Also, note that when entering comments into a domain file for a dynamic zone, the comments will be deleted when the first update to the domain is made. Domain file comments are not maintained because they would degrade the performance of the file update process.

Change the configuration information for a dynamic domain in two different ways:

- Manually, by editing the domain files *only after shutting down the DDNS server*
- Dynamically, by using **nsupdate** while the DDNS server is running

For information on how to manually enter configuration information into domain files, refer to “Step 5. Create the domain data files (master name server only)” on page 433.

When first setting up your BIND 4.9.3 dynamic domain, **nsupdate -g** was used to create the zone key RSA key pair; **nsupdate -g** creates an entry in the /etc/ddns.dat file. The public key, the second key in the /etc/ddns.dat file entry, needs to be copied into the appropriate domain file. The zone private key stored in /etc/ddns.dat is used by **nsupdate** when signing update requests for the administrator of the zone. The server then examines the signature to identify update requests from the zone administrator versus those from ordinary hosts. The zone key gives the possessor the power to use **nsupdate** to create, modify, and delete any host’s record in a dynamic domain.

Once the zone key information is generated and the DDNS server started, the administrator can take the /etc/ddns.dat file with him and administer the zone remotely using **nsupdate**.

Steps for Migrating an existing DNS configuration to BIND 4.9.3 dynamic IP

Before you begin: You need to decide if you want to:

- Leave existing resource records as they are and allow new ones to be created and updated dynamically. This will allow existing systems to keep their host names, but they will not be able to update their resource records dynamically unless a system administrator deletes them.
- Delete all existing resource records and start with a dynamic domain from the beginning.

Perform the following steps to migrate existing DNS server configuration files to Dynamic IP:

1. For a boot file, add the *dynamic secured* or *dynamic presecured* keywords to the *primary* statements for the authoritative DNS server that is being upgraded.

   _____

2. Use the NSUPDATE -g command to create the encryption keys. Copy the public key to the zone files.

   _____

3. Start the DDNS server.

   _____

4. Make a new entry in the DDNS.DAT file for each zone the DHCP server will update, if you have a DHCP server configured for DDNS updates.

   _____

You have now migrated existing DNS server configuration files to Dynamic IP.

RSA encryption

Because the z/OS UNIX DDNS server and client products implement not only dynamic DNS but also DNS security functions, below is a brief explanation of cryptography. This section is courtesy of RSA Data Security, Inc., Redwood City, California, and has been modified slightly here for z/OS CS.

Secret key cryptography: This method uses a secret key to encrypt a message. The same secret key must be used again to decrypt the message. This means that the key must be sent along with the message which exposes it to whoever might be eavesdropping on the conversation. Secret keys are very fast in terms of processing, and it is not easy to break them even though they are exposed through the communication process.

Public key cryptography: This method uses a combination of a modulus and a pair of exponents, called the public key and the private key. Exponents and modulus must be used together to encrypt or decrypt a message, but only the modulus and the public exponent are communicated since they are important to everyone who wants to send or receive encrypted messages using this method. The private exponent will never be publicly exposed. This ensures that no one else can decrypt messages that have been intended for a specified recipient, nor can anyone else disguise as that recipient to intercept a message.

Encryption and authentication: Encryption means that a message will be scrambled before it can be sent over a communications link. The plain message itself will never be sent to ensure privacy. Authentication is used to ensure that a message has indeed originated from the source specified in the message, and that the message has not been altered in transit. It additionally serves the purpose of non-repudiation, which means that whoever has digitally signed a message cannot claim later that he or she has not done so. In this case, the plain message itself will be sent since there is no need for privacy. The message will also be used to generate a digital signature by using one of the aforementioned cryptographic methods, preferably public keys.

Hash functions: A hash function is a computation that takes a variable-size input and returns a fixed-size string, which is called the hash value. If the hash function is one-way, that means hard to invert, it is also called a message-digest function, and the result is called a message digest. The idea is that a digest represents concisely the longer message or document from which it was computed; one can think of a message digest as a digital fingerprint of the larger document.

The RSA encryption standard: This standard public key encryption method, along with the MD5 hash function, is used with the IBM DDNS product in z/OS CS. The principle of the RSA algorithm is as follows:

1. Take two large primes, p and q.
2. Find their product n = p * q; n is called the modulus.
3. Choose a number, e, less than n and relatively prime to (p-1) * (q-1).
4. Find its inverse, d, mod (p-1) * (q-1), which means that e * d = 1 mod (p-1) * (q-1).

e and d are called the public and private exponents, respectively. The public key is the pair (n,e); the private key is d. The factors p and q must be kept secret or destroyed.

An example of RSA privacy (encryption) follows. Suppose Alice wants to send a private message, m, to Bob. Alice creates the ciphertext c by exponentiating:

```
c = m^e mod n
```

where e and n are Bob's public key. To decrypt, Bob also exponentiates:

```
m = c^d mod n
```

and recovers the original message, m; the relationship between e and d ensures that Bob correctly recovers m. Since only Bob knows d, only Bob can decrypt.

An example of RSA authentication follows. Suppose Alice wants to send a signed document, m, to Bob. Alice creates a digital signature s by exponentiating:

```
s = m^d mod n
```

where d and n belong to Alice's key pair. She sends s and m to Bob. To verify the signature, Bob exponentiates and checks that the message, m, is recovered:

```
m = s^e mod n
```

where e and n belong to Alice's public key.

Thus encryption and authentication take place without any sharing of private keys: each person uses only other people's public keys and his or her own private key. Anyone can send an encrypted message or verify a signed message, using only public keys, but only someone in possession of the correct private key can decrypt or sign a message.

To make encryption methods secure, a fairly large modulus should be chosen since it becomes increasingly difficult to break a large number into factors to determine the original primes. RSA uses a minimum length of 512 bits for the modulus, which would convert to a number with approximately 155 digits.

Due to security concerns, public key systems that use a key length of more than 512 bits must not be exported from the US.

For encryption, in reality, RSA is combined with a secret-key crypto system, such as DES, to encrypt a message by means of an RSA digital envelope. Data Encryption Standard (DES) is one of the most widely used secret key algorithms and was originally developed by IBM.

Suppose Alice wishes to send an encrypted message to Bob. She first encrypts the message with DES, using a randomly chosen DES key. Then she looks up Bob's public key and uses it to encrypt the DES key. The DES-encrypted message and

the RSA-encrypted DES key together form the RSA digital envelope and are sent to Bob. Upon receiving the digital envelope, Bob decrypts the DES key with his private key, then uses the DES key to decrypt the message itself.

For authentication, in reality, RSA is combined with a hash function, such as MD5.

Suppose Alice wishes to send a signed message to Bob. She uses a hash function on the message to create a message digest, which serves as a digital fingerprint of the message. She then encrypts the message digest with her RSA private key; this is the digital signature, which she sends to Bob along with the message itself. Bob, upon receiving the message and signature, decrypts the signature with Alice's public key to recover the message digest. He then hashes the message with the same hash function Alice used and compares the result to the message digest decrypted from the signature. If they are exactly equal, the signature has been successfully verified, and he can then be confident that the message did indeed come from Alice. If, however, they are not equal, then the message either originated elsewhere or was altered after it was signed, and he rejects the message.

For authentication, the roles of the public and private keys are converse to their roles in encryption, where the public key is used to encrypt and the private key to decrypt. In practice, the public exponent is usually much smaller than the private exponent; this means that the verification of a signature is faster than the signing. This is recommended because a message or document will only be signed by an individual once, but the signature can be verified many times.

Configuring the DHCP server for z/OS

Dynamic Host Configuration Protocol (DHCP) allows clients to obtain IP network configuration information, including IP addresses, from a central DHCP server. The DHCP server controls whether the addresses it provides to clients are allocated permanently or are leased for a specific period. When a client is allocated a leased address, it must periodically request that the server revalidate the address and renew the lease.

The process of address allocation, leasing, and lease renewal are all handled dynamically by the DHCP client and server programs and are transparent to the end user.

DHCP defines three IP address allocation policies:

Dynamic
A DHCP server assigns a temporary, leased IP address to a DHCP client.

Static A DHCP server administrator assigns a static, predefined address reserved for a specific DHCP client.

Permanent
A DHCP server administrator assigns a permanent IP address to a DHCP client. No process of lease renewal is required.

Note: If the network uses routers or gateways, ensure that they can be enabled as DHCP relay agents. Enabling the routers or gateways for DHCP allows the DHCP packets to be sent across the network to other LAN segments.

If there are no routers that can configure to be used as DHCP relay agents, you could:

- Use a UNIX system or RS/6000® system that has the necessary code to be configured to receive limited DHCP broadcasts. Then, forward those broadcast requests to the appropriate host server.
- Use a host server that is located on the same LAN segment as the IBM Network Stations. This would eliminate any need for routers or intermediate UNIX systems to pass on the broadcast requests of the IBM Network Stations.

For dynamic address allocation, a DHCP client that does not have a permanent lease must periodically request the renewal of its lease on its current IP address in order to keep using it. The process of renewing leased IP addresses occurs dynamically as part of the DHCP and is transparent to the user.

How does DHCP work?: DHCP allows clients to obtain IP network configuration information, including IP addresses, from a central DHCP server. DHCP servers control whether the addresses they provide to clients are allocated permanently or are *leased* for a specific time period. When a client receives a leased address, it must periodically request that the server revalidate the address and renew the lease.

The DHCP client and server programs handle the processes of address allocation, leasing, and lease renewal.

To further explain how DHCP works, look at some frequently asked questions:

- How is configuration information acquired?
- How are leases renewed?
- What happens when a client moves out of its subnet?
- How are changes implemented in the network?

Acquiring configuration information: DHCP allows DHCP clients to obtain an IP address and other configuration information through a request process to a DHCP server. DHCP clients use RFC-architected messages to accept and use the options served them by the DHCP server. For example:

1. The client broadcasts a message (containing its client ID) announcing its presence and requesting an IP address (DHCPDISCOVER message) and desired options such as subnet mask, domain name server, domain name, and static route.
2. Optionally, if routers on the network are configured to forward DHCP and BOOTP messages (using BOOTP Relay), the broadcast message is forwarded to DHCP servers on the attached networks.
3. Each DHCP server that receives the client's DHCPDISCOVER message can send a DHCPOFFER message to the client offering an IP address (a server that does not want to serve a client can simply ignore the DHCPDISCOVER).

   The server checks the configuration file to see if it should assign a static or dynamic address to this client.

   In the case of a dynamic address, the server selects an address from the address pool, choosing the least recently used address. An address pool is a range of IP addresses to be leased to clients. In the case of a static address, the server uses a Client statement from the DHCP server configuration file to assign options to the client. Upon making the offer, the IBM DHCP server reserves the offered address.
4. The client receives the offer messages and selects the server it wants to use.

5. The client broadcasts a message indicating which server it selected and requesting use of the IP address offered by that server (DHCPREQUEST message).
6. If a server receives a DHCPREQUEST message indicating that the client has accepted the server's offer, the server marks the address as leased. If the server receives a DHCPREQUEST message indicating that the client has accepted an offer from a different server, the server returns the address it offered to the client to the available pool. If no message is received within a specified time, the server returns the address it offered to the client to the available pool. The selected server sends an acknowledgment which contains additional configuration information to the client (DHCPACK message).
7. The client determines whether the configuration information is valid. Upon receipt of a DHCPACK message, the DHCP client sends an Address Resolution Protocol (ARP) request to the supplied IP address to see if it is already in use. If it receives a response to the ARP request, the client declines (DHCPDECLINE message) the offer and initiates the process again. Otherwise, the client accepts the configuration information.
8. Accepting a valid lease, the client enters a BINDING state with the DHCP server, and proceeds to use the IP address and options.

To DHCP clients that request options, the DHCP server typically provides options that include subnet mask, domain name server, domain name, static route, class-identifier (which indicates a particular vendor), user class, and the name and path of the load image.

However, a DHCP client can request its own, unique set of options. For example, Windows NT 3.5.1 DHCP clients are required to request options. The default set of client-requested DHCP options provided by IBM includes subnet mask, domain name server, domain name, and static route. For option descriptions, see "Specifying DHCP options" on page 520.

Renewing leases: The DHCP client keeps track of how much time is remaining on the lease. At a specified time prior to the expiration of the lease, usually when half of the lease time has passed, the client sends a renewal request, containing its current IP address and configuration information, to the leasing server. If the server responds with a lease offer, the DHCP client's lease is renewed.

If the DHCP server explicitly refuses the request, the DHCP client might continue to use the IP address until the lease time expires and then initiate the address request process, including broadcasting the address request. If the server is unreachable, the client might continue to use the assigned address until the lease expires.

Moving a client out of its subnet: One benefit of DHCP is the freedom it provides a client host to move from one subnet to another without having to know ahead of time what IP configuration information it needs on the new subnet. As long as the subnets to which a host relocates have access to a DHCP server, a DHCP client will automatically configure itself correctly to access those subnets.

For a DHCP client to reconfigure itself to access a new subnet, the client host must be rebooted. When a host restarts on a new subnet, the DHCP client might try to renew its old lease with the DHCP server which originally allocated the address. The server refuses to renew the request since the address is not valid on the new subnet. Receiving no server response or instructions from the DHCP server, the client initiates the IP address request process to obtain a new IP address and access the network.

Implementing changes in the network: With DHCP, you can make changes at the server, reinitialize the server, and distribute the changes to all the appropriate clients. A DHCP client retains DHCP option values assigned by the DHCP server for the duration of the lease. If you implement configuration changes at the server while a client is already up and running, those changes are not processed by the DHCP client until the client attempts to renew its lease or until it is restarted.

Setting up a DHCP network: The following sections contain information to help you in setting up your DHCP system:

- To create a scoped DHCP network, see "Creating a scoped network".
- To start the DHCP server, see "Starting the DHCP server" on page 505.
- For tips on maintaining a DHCP server, see "Maintaining the DHCP server" on page 505.

The IBM DHCP server provides configuration information to clients based on statements contained in the server's configuration file and based on information provided by the client. The server's configuration file defines the policy for allocating IP addresses and other configuration parameters. The file is a *map* that the server uses to determine what information should be provided to the requesting client.

Before starting the DHCP server, create or modify the DHCP server configuration file.

Once the DHCP server is running, you can also make dynamic changes to the configuration by modifying the configuration file and using the DHCP Server Maintenance program to reinitialize the DHCP server.

Creating a scoped network: You create a hierarchy of configuration parameters for a DHCP network by specifying some configuration values that are served globally to all clients, while other configuration values are served only to certain clients. Serving different configuration information to clients is often based on network location, equipment vendor, or user characteristics.

Depending on your configuration, you can specify subnets, classes, vendors, and clients to provide configuration information to different groups of clients:

- When defined globally, client, vendor or class options are available to DHCP clients regardless of their network location.

  Parameters specified for a subnet, class, or client are considered local to the subnet, class, or client. A client defined within a subnet inherits both the global options and the options defined for that subnet. If a parameter is specified in more than one level in the network hierarchy, the lowest level (which is the most specific) is used.
- Use the Subnet statement to specify configuration parameters for one subnet for a specific location in your network or enterprise.
- Use the Class statement to configure DHCP classes to provide unique configuration information from the server to clients that identify themselves as belonging to that class. For example, a group of clients can all use a shared printer or load image.
- Use a Vendor statement to provide unique configuration information to clients that identify themselves as using a specific vendor's equipment or software. Specially defined options can be served to these clients.
- Use a Client statement in the DHCP server configuration file to serve specified options to a specific client or to exclude that client from service. You can also use a Client statement to exclude IP addresses from service.

For more information on obtaining information for a DHCP client, see “Maintaining the DHCP server”.

Handling errors in configuration files: Configuring the server incorrectly causes few, if any, warning messages. The DHCP server normally runs even when it encounters errors in the configuration file. The server might ignore the incorrect data and optionally post a message to its log.

Starting the DHCP server:

Note: DHCPSD is installed in the /usr/lpp/tcpip/sbin directory.

To start the DHCP server, use a start procedure (found in the EZATDHDP member of SEZAINST), or the following form of the **dhcpsd** command:

dhcpsd [-q|-v] [-f configFile]

-q Starts the server in **quiet** mode, which means that no banner is displayed when the server starts.

-v Starts the server in **verbose** mode. Causes messages dealing with client communication to print to screen.

-f configFile
Is the name of the DHCP server configuration file. By default, the server searches for a file called dhcpsd.cfg in the directory specified by the ETC environment variable.

When starting the DHCP server with a procedure (proc), the example start proc is found in the DHCP member of the installation partitioned data set SEZAINST.

Maintaining the DHCP server:

Note: DADMIN is installed in the /usr/lpp/tcpip/sbin directory.

To maintain a running DHCP server, IBM provides the **dadmin** command to:

- Reinitialize a DHCP server by causing the server to reread its configuration file
- Delete a lease
- Control server tracing
- Display client information
- Display IP address information
- Display server statistics

Note: Verbose mode provides additional information for debugging purposes. Verbose mode is allowed on any of the following **dadmin** command instances. Verbose is shown as a parameter in those instances where additional, more detailed information is of particular value.

Displaying dadmin command syntax: To display information about the command syntax, enter:

dadmin -?

Reinitializing the running server: If you make changes to the configuration file, you will need to reinitialize the running server to implement the changes. To reinitialize the server, use the following form of the **dadmin** command:

dadmin [[**-h**]*host*] **-i** [**-v**]

-h Specifies the host

host
The IP address or host name of the DHCP server. If no server is specified, the local server is assumed.

-i Reinitializes the specified server.

-v Executes the command in verbose mode.

Displaying client information: To display information for a client ID, use the following form of the **dadmin** command:

dadmin -c*value* [**-v**]

-c Requests information for one or more clients that match this client ID.

value
The client ID is a MAC address. For example, enter 004ac77150fc. Information is returned for any matching hardware type.

-v Executes the command in verbose mode.

Displaying IP address information: To display information for one IP address, use the following form of the **dadmin** command:

dadmin -q*n.n.n.n* [**-v**]

-q Requests the IP address information.

n.n.n.n
The IP address of the client.

-v Executes the command in verbose mode.

Information returned is an address record for the IP address:

- IP Address - IP address
- Status - Status of the IP Address
  - N/A - Address is not available
  - Free - Address is available
  - Reserved - Address is available, but is reserved for a specific client
  - Leased - Address is currently leased to a client
  - Released - Address has been released by the client
  - Expired - Address lease has expired
  - Used - Address is not available because it is in use in the network
- Lease Time - Length of current lease
- Start Time - Time when address was first leased
- Last Leased - Time of most recent lease
- Proxy - DNS A record update done for the client when lease obtained.
- Client ID - ID for the client associated with this IP address

Querying an address pool: To display information for a pool of IP addresses, use the following form of the **dadmin** command:

dadmin -p*n.n.n.n* [**-v**]

-p Requests the address pool information.

n.n.n.n
The IP address of the address pool.

-v Executes the command in verbose mode.

Information returned is an address record for each IP address in the pool:

- IP Address - IP address
- Status - Status of the IP Address
  - N/A - Address is not available
  - Free - Address is available
  - Reserved - Address is available, but is reserved for a specific client
  - Leased - Address is currently leased to a client
  - Released - Address has been released by the client
  - Expired - Address lease has expired
  - Used - Address is not available because it is in use in the network
- Lease Time - Length of current lease
- Start Time - Time when address was first leased
- Last Leased - Time of most recent lease
- Proxy - DNS A record update done for the client when lease obtained.
- Client ID - ID for the client associated with this IP address

Querying all address pools: To display information for all IP addresses, use the following form of the **dadmin** command:

dadmin -s [**-v**]

-s Requests the address information for all IP addresses.

-v Executes the command in verbose mode.

Returns address records (as described above) for all addresses in the DHCP server's address pools.

Controlling server tracing: To start and stop tracing on the DHCP server, use the following form of the **dadmin** command:

dadmin -t*value* [**-v**]

-t Specifies server tracing.

value
The value is ON to start tracing or OFF to stop tracing.

-v Executes the command in verbose mode.

Displaying server statistics: To display statistics information about the pool of addresses administered by the server, use the following form of the **dadmin** command:

dadmin [[**-h**]*host*] **-n***value* [**-v**]

-h Specifies the host

host
The IP address of the DHCP server. If no host is specified, the local server is assumed.

-n Requests statistics for the server specified as *host*.

value
The value is a decimal integer indicating the number of intervals from 0 to 100. For example, a value of three returns a summary record that includes totals information, the current interval record, and the 3 most recent history records. A value of 0 returns a summary record of activity since the last summary.

-v Executes the command in verbose mode.

Statistics include:

- Discover packets processed
- Discover packets with no response
- Offers made
- Leases granted
- Negative acknowledgments (NAKs)
- Informs processed, including informs plus acknowledgments (ACKs)
- Renewals
- Releases
- BOOTP clients processed
- proxyARec updates attempted
- Unsupported packets
- Monitor requests processed

Deleting leases: If you find that an assigned lease is not being used and you want to make the IP address available for allocation, you can delete the lease. You can only delete one lease at a time. You will be prompted to confirm deletion of the lease. To delete the lease, use the following form of the **dadmin** command:

dadmin [**-f**] [**-v**] [[**-h**]*host*] **-d** *ip_address*

-f Forces deletion of the lease without prompting.

-v Executes the command in verbose mode.

-h

host
Specifies the IP address of the DHCP server. If no server is specified, the local server is assumed.

-d Deletes the lease for the specified IP address.

ip_address
The IP address for the lease to be deleted.

Configuring the DHCP server for the IBM Network Station® client: You can configure the DHCP server to be used by an IBM Network Station. The DHCP server sets up the subnet and specifies the next bootstrap server. The IBM Network Station client can request information. The DHCP server should be configured to provide options that include subnet mask, router, domain name, and boot file name.

Changing the DHCP configuration file

The name of the DHCP server configuration file is dhcpsd.cfg. The default location for dhcpsd.cfg is the \etc directory. In the configuration file, you create a hierarchy of configuration parameters for a DHCP network by specifying some configuration values that are served globally to all clients and other configuration values that are served only to certain clients. The information supplied to the clients is determined by the statements you use and the position of the statements in the configuration file.

Depending on your configuration, you can specify subnets, classes, vendors, and clients to provide configuration information to different groups of clients:

- When defined globally, client, vendor or class options are available to DHCP clients regardless of their network location.

  Parameters specified for a subnet, class, or client are considered local to the subnet, class, or client. A client defined within a subnet inherits both the global options and the options defined for that subnet. If a parameter is specified in more than one level in the network hierarchy, the lowest level (which is the most specific) is used.
- Use the Subnet statement to specify configuration parameters for one subnet for a specific location in your network or enterprise.
- Use the Class statement to configure DHCP classes to provide unique configuration information from the server to clients that identify themselves as belonging to that class. For example, a group of clients can all use a shared printer or load image.
- Use a Vendor statement to provide unique configuration information to clients that identify themselves as using a specific vendor's equipment or software. Specially-defined options can be served to these clients.
- Use a Client statement in the DHCP server configuration file to serve specified options to a specific client or to exclude that client from service. You can also use a Client statement to exclude IP addresses from service.
- A sample DHCP server configuration file is installed in the HFS as /usr/lpp/tcpip/samples/dhcpsd.cfg.

Editing tips: When editing the DHCP server configuration file, keep in mind the following:

- Comments must begin with a # character.
- Class and vendor names that include spaces must be surrounded by double quotes (″).
- Statement parameters are dependent on their position. If you omit a required parameter and enter a subsequent required parameter in a statement, the server recognizes that a parameter is missing, writes an error message to a log file, and continues to read the configuration file.
- A continuation character \ indicates that the information is continued on the next line. When used within a comment, the character is treated as part of the comment and is ignored as a continuation character.
- Braces are used to specify statements that are defined within other statements.
- If a parameter is specified in more than one place, the lowest level statement (which is the most specific) is used:
  - Statements specified outside braces are considered global and are used for all addresses served by this server unless the statement is overridden at a lower-defined level.

  - Parameters specified within braces under a statement such as a Subnet statement are considered local and apply only to clients within the subnet.
  - Definition of a parameter in a class takes precedence over definition of the parameter in a subnet.
- Class statements are not allowed inside Client statements.
- Client statements are not allowed inside Option, Vendor, or Class statements.
- Subnet statements are not allowed inside Class or Client statements.
- Vendor statements are always defined at a global level.
- Keywords are not case-sensitive. The capitalization that is used in this documentation is not required in the configuration file. However, use the convention that keywords start with a lowercase letter and subsequent word subparts start with a capital letter. For example, a keyword is proxyARec.

DHCP server statements:

ServerType statement: To specify whether the server will operate only as a standard DHCP server, will perform both normal DHCP operations and PXE proxy DHCP operations, or will act only as a redirection server (PXE proxy DHCP server), use the following statement:

```
ServerType [DHCP | PXEDHCP | PXEPROXY]
```

The default value is `DHCP`, meaning the server will operate only as a standard DHCP server and will not interpret any PXE client extensions. The server will, however, still pass DHCP options and parameters to PXE clients.

PXEDHCP specifies that the server will perform both normal DHCP operations and PXE proxy DHCP operations. PXEPROXY indicates that the server will act only as a redirection server (PXE proxy DHCP server).

ImageServer statement: To specify the siaddr field of the reply packet for a server acting as a PXE DHCP or PXE proxy only server, use the following statement:

```
ImageServer [ip address | hostname]
```

This statement separates the PXE siaddr field (the address of the BINL server) from the DHCP siaddr field. If the server is acting as a PXE DHCP server, it will use the ImageServer value to fill in the siaddr field of the reply packet. If the ImageServer statement is not specified, the siaddr field is left null for the PXE packet. A null value indicates that the BINL server resides on the same machine as the PXE proxy server. ImageServer is a global statement.

Log statements: To enable logging by the server, use the following statements:

- Number of DHCP log files.

  `numLogFiles` *`number_of_log_files`*

  number_of_log_files is the maximum number of log files maintained.
- Size of DHCP log file.

  `logFileSize` *`size_of_log_file`*

  size_of_log_file is the size of the log file in kilobytes.
- Name of DHCP log file.

  `logFileName` *`name_of_log_file`*

  name_of_log_file is the name of the most recent log file.

- Type of log item.

  `logItem` *type_of_log_item*

  type_of_log_item is the type of the item to be logged. You should specify at least one log item. *type_of_log_item* can be:

  ```
  SYSERR
  OBJERR
  PROTERR
  WARNING
  EVENT
  ACTION
  INFO
  ACNTING
  TRACE
  ```

supportBootP statement: To specify whether the server responds to requests from BOOTP clients, use the following statement:

```
supportBootP [YES | NO]
```

The default value is `NO`.

If this server previously supported BOOTP clients and has been reconfigured not to support BOOTP clients, the address binding for any BOOTP clients that was established before the reconfiguration is maintained until the BOOTP client sends another request (when it is restarting). At that time, the server does not respond, and the binding is removed.

Use this statement outside of braces.

supportUnlistedClients statement: To specify whether the server responds to requests from DHCP clients other than those whose client IDs are specifically listed in this configuration file, use the following statement:

```
supportUnlistedClients [YES | NO]
```

The default is YES. If you specify NO, the server responds only to requests from DHCP clients that are listed (by client ID) in the configuration file.

For example:

```
client 6 10005aa4b9ab ANY
client 6 10a03ca5a7fb ANY
```

If the `supportUnlistedClients` statement is not specified or if you specify YES, the server responds to requests from any DHCP client. Use this option to limit access to addresses that are issued by this DHCP server. Listing the client IDs for all acceptable clients might take a long time.

Use this statement outside of braces.

bootStrapServer statement: To specify whether the DHCP server specifies a bootstrap server for BOOTP clients, use the following statement:

`bootStrapServer` *address_of_bootstrap*

address_of_bootstrap is the IP address of the bootstrap server for the client.

This statement can appear at the global level, or within a Subnet, Class, or Client statement.

Option 67, Boot File Name: For clients who need a boot or must load images to initialize, use the DHCP Option 67 (Boot File Name) for the name of the boot file. The client downloads the image from the BOOTP server. For additional information about Option 67, see "Specifying DHCP options" on page 520.

Lease statements:

- To specify the default lease duration for the leases that are issued by the server, use the following statement:

  `leaseTimeDefault` *amount_of_default_lease_time*

  amount_of_default_lease_time is a decimal integer, followed by a space and a unit of time. The unit of time can be years, months, weeks, days, hours, minutes, or seconds.

  Default interval:
  24 hours (1440 minutes)

  Default unit:
  minute

  Minimum:
  180 seconds

  Maximum:
  -1, which is infinity

  You can use this statement at the global level. To override this statement for a set of clients, use Option 51 (IP Address Lease Time) for a specific client, a class of clients, a subnet, or at the global level.
- To specify the interval at which the lease condition of all addresses in the address pool is examined, use:

  `leaseExpireInterval` *interval_of_lease_time*

  interval_of_lease_time is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. If you do not specify a unit of time, minutes are assumed. The value that is specified should be less than the value for `leaseTimeDefault` to ensure that expired leases are returned to the pool in a timely manner.

  Default interval:
  1 minute

  Default unit:
  minute

  Minimum:
  15 seconds

  Maximum:
  12 hours
- To specify the maximum amount of time the server holds an offered address in reserve while waiting for a response from the client, use:

  `reservedTime` *amount_of_time_reserved*

  amount_of_time_reserved is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. If you do not specify a unit of time, minutes are assumed.

Default interval:
5 minutes

Default unit:
minute

Minimum:
30 seconds

Maximum:
-1, which is infinity

- To improve the performance of DHCP, limit the times the server accesses the dhcps.ar and dhcps.cr DHCP database files by using the DHCP smart caching feature. Use this feature to reduce the chance of not receiving a DHCP server response to a client request.

  Smart caching can help in cases where the dhcps.ar and dhcps.cr files are large. Smart caching works with the leaseExpireInterval. The default interval is one minute and the maximum interval is 12 hours. Smart caching uses this value as the time interval to access and update dhcps.ar and dhcps.cr files. During this time interval, the DHCP server keeps the new incoming data in memory. The value setting is dependent on the system stability.

  If the DHCP server ends before the leaseExpireInterval, the new data may be lost. Smart caching is enabled at DHCP initialization if the file /etc/dhcp_sc exists in the HFS. The file content is not important, only the existence of the file itself. If the file is located, a DHCP log file message will state this fact and smart caching will be activated.

pingTime statement: For each DHCP client request the DHCP server issues a ping to determine if the IP address chosen for the client is already in use. The server waits the specified time for a ping response before assuming that the IP address is not in use.

```
pingTime  n.n second
```

The value is a time amount. The default time is one second. The use of floating point numbers for the time amount is allowed such as 0.1 seconds to set the ping time to a tenth of a second. The maximum value is 30 seconds. If the amount is set to 0 the DHCP server will not issue a ping.

Subnet statement: Use the Subnet statement to specify configuration parameters for an address pool that is administered by a server. An address pool is a range of IP addresses to be leased to clients. If you configure subnets, you can set the lease time and other options for clients that use the address pool.

- Define a subnet.

  To define a subnet, use the following statement:

  ```
  subnet subnet_address [subnet_mask] subnet_range [(alias=subnet_name ]
  ```

  Note: The DHCP Server Configuration program uses the parameters to the right of a left parenthesis. The DHCP server parses statements to the right of a left parenthesis as comments.

  subnet_address
  The address of this subnet, specified in dotted-decimal notation (for example, 192.67.48.0).

  subnet_mask
  The mask for the subnet, specified in dotted decimal notation or in integer format. A subnet mask divides the subnet address into a subnet portion and

a host portion. If no value is entered for the subnet mask, the default is the class mask appropriate for an A, B, or C class network.

– Class A network - 255.0.0.0
– Class B network - 255.255.0.0
– Class C network - 255.255.255.0

A subnet mask can be expressed either in dotted-decimal notation or as an integer between 8 and 31. For example, you can enter a subnet mask as a dotted decimal notation of 255.255.240.0 or an integer format of 20. In subnet 192.67.48.0, a mask of 255.255.240.0 implies an address range from 192.67.48.001 to 192.67.63.254. The value 20 is the total number of 1's in a mask that is expressed in binary as 11111111.11111111.11110000.00000000.

Although not required, in most configurations the DHCP server should send Option 1 (Subnet Mask) to DHCP clients. Client operation can be unpredictable if the client does not receive subnet masks from the DHCP server and assumes a subnet mask that is not appropriate for the subnet.

subnet_range
All addresses, specified in dotted-decimal notation, to be administered to this subnet. The ranges should not overlap, for example, 192.67.48.1-192.67.48.128.

Notes:

1. In the range of addresses, do not include the address of the subnet and the address that is used for broadcast messages. For example, if the subnet address is 192.67.96.0 and the subnet mask is 255.255.240.0, do not include 192.67.96.0 and 192.67.111.255 in the range of addresses.
2. To exclude an IP address in a range of address, use the Client statement (see step 517).

(alias=*subnet_name*
A symbolic name for ease in identifying a subnet.

- Define a subnet group.

  To define a subnet group, use the following statement:

  ```
  subnet subnet_address [subnet_mask] subnet_range [label:value[/priority]]
  ```

  label:*value[/priority]*
  Identifies subnets that are grouped together on the same wire. *value[/priority]* is a string of 1 to 64 alphanumeric characters that identifies the subnet, followed by the priority in which this subnet's address pool is used. Do not uses spaces when specifying the label. More than one subnet can have the same identifier. *priority* is a positive integer, where 1 is a higher priority than 2. If you do not specify a priority, the highest priority is assigned. If two subnets have identical priority, the subnets within a label are processed based on the physical position in the configuration file.

  For example, the following two subnets are on the same wire:

  ```
  inOrder
  subnet 192.67.49.0 255.255.240.0 192.67.49.1-192.67.49.100 label:WIRE1/2
  subnet 192.67.48.0 255.255.240.0 192.67.48.1-192.67.48.50 label:WIRE1/1
  ```

- Serve IP addresses from multiple subnets.

  To serve IP addresses from multiple subnets, use the `inOrder` or `balance` statement. The `inOrder` or `balance` statement is defined at a global level.

  inOrder *subnet_labelslist*
  subnet_labelslist is a list of labels in which each label identifies a subnet

group. Each listed group is processed in order within that group. The subnet address pool with the highest priority within that group is completely exhausted before the subnet address pool with the next highest priority is used.

balance: *subnet_labelslist*
subnet_labelslist is a list of labels in which each label identifies a subnet group. The server provides the first IP address from the subnet that is first in the priority list, and subsequent IP addresses from each lesser-priority subnet, repeating the cycle until addresses are exhausted equally from all subnets.

The following is an example using the `inOrder` statement. Requests for subnet group WIRE1 exhaust addresses in subnet 192.67.48.0 (WIRE1/1) first, followed by subnet 192.67.49.0 (WIRE1/2). WIRE1 and WIRE3 are not related. Requests for subnet group WIRE3 exhaust addresses in subnet 192.67.50.0 (WIRE3/1) first, followed by subnet 192.67.51.0 (WIRE3/2), and then 192.67.50.0 (WIRE3/3), which has the same subnet address as WIRE3/1 but specifies a higher address range:

```
inOrder: WIRE3 WIRE1
subnet 192.67.49.0 255.255.240.0 192.67.49.1-192.67.49.100 label:WIRE1/2
subnet 192.67.48.0 255.255.240.0 192.67.48.1-192.67.48.50 label:WIRE1/1
subnet 192.67.51.0 255.255.240.0 192.67.51.1-192.67.51.50 label:WIRE3/2
subnet 192.67.50.0 255.255.240.0 192.67.50.1-192.67.50.50 label:WIRE3/1
subnet 192.67.50.0 255.255.240.0 192.67.50.51-192.67.50.100 label:WIRE3/3
```

The following `balance` statement uses all IP addresses equally in WIRE1/3 and WIRE1/4:

```
balance: WIRE1
subnet 192.67.49.0 255.255.240.0 192.67.49.101-192.67.49.200 label:WIRE1/3
subnet 192.67.48.0 255.255.240.0 192.67.48.201-192.67.48.300 label:WIRE1/4
```

A sequence of `inOrder` or `balance` statements is cumulative. For example, the statements:

```
inOrder: WIRE1
inOrder: WIRE3
```

have the cumulative effect of the single statement:

```
inOrder: WIRE1 WIRE3
```

Note: To disable multiple subnets, comment out either the `balance` or `inOrder` processing statement or the priority.

Class statement: Use the Class statement to specify configuration parameters for a user-defined group of clients that are administered by a server. You can use the Class statement at the global or subnet level. When the Class statement is specified within a Subnet statement, the server only serves clients in the class that are located in the specified subnet and request the class.

For example, to create a class that is called ″accounting″ that allows member hosts to use the LPR server (Option 9) at 192.67.123.2, do the following:

- At the DHCP server, define a class that is called ″accounting″ and set the LPR server for that class to 192.67.123.2
- At the client, configure the client to identify itself as belonging to the class ″accounting″.

When the client requests configuration information, the server sees that it belongs to the accounting class and provides configuration information that instructs the client to use the LPR server at 192.67.123.2. DHCP clients use Option 77 to indicate their class to DHCP servers.

To define a class, use the following statement:

```
class class_name [class_range]
```

class_name
The user-defined label that identifies the class. The class name is an ASCII string of up to 255 characters (for example, accounting). If the class name contains spaces, it must be surrounded by double quotes.

class_range
A range of addresses. To specify a range of addresses, enter addresses in dotted-decimal notation, beginning with the lower end of the range, followed by a hyphen, then the upper end of the range, with no spaces between. For example, enter 192.17.32.1-192.17.32.128.

At a global level, a class cannot have a range. A range is allowed only when a class is defined within a subnet. The range can be a subset of the subnet range.

A client that requests an IP address from a class that has used all the addresses from its range is offered an IP address from the subnet range, if available. The client is offered the options associated with the class that has used all the addresses from its range.

To assign configuration parameters such as a lease time for all clients in a class, follow the Class statement with Option statements that are surrounded by braces. For more information about options, see“Specifying DHCP options” on page 520.

Client statement: Use the Client statement to specify a unique set of options for a client. You can assign either a static address and configuration parameters, or configuration parameters. You can also use the Client statement to exclude an IP address from a range of available IP addresses. You can use the Client statement at the global, subnet, or class level.

- Define a client.

  `client` *hw_type clientID client_ipaddr* (`alias=`*client_name*

  hw_type
  A number that represents the hardware type of the client computer, required to decode the MAC address. For more information about hardware types, see “Hardware types” on page 521.

  clientID
  Either the hexadecimal MAC address, a string such as a domain name, or a name that is assigned to the client such as the host name. If you specify a string, you are required to enclose it in double quotes and specify 0 for the hardware type.

  client_ipaddr
  The DHCP client’s IP address in dotted-decimal notation. *client_ipaddr* must contain an address if unlisted clients are not supported.

  (alias=*client_name*
  A symbolic name for ease in identifying the client. Enter **alias=***client_name*

immediately after a left parenthesis. This symbolic name appears in the display of the server configuration. If no name is entered, the MAC address is used.

For example, for a client (10005aa4b9ab), to reserve the static address 192.22.3.149 and also specify a lease time (Option 51) or 12 hours (43200 seconds) and a subnet mask (Option 1), use the following statement:

```
client 6 10005aa4b9ab 192.22.3.149
{
   option 51 43200
   option 1 255.255.255.0
}
```

- Specify options and assign any IP address to a client.

  To specify options for any IP address from the subnet, use the following `client` statement:

  `client` *hw_type clientID* `ANY`

  ANY
  Specifies that any IP address can be assigned to the specific client ID.

- Exclude a client ID.

  To have the DHCP server exclude requests from a particular client ID, use the following `client` statement:

  `client` *hw_type clientID* `NONE`

  NONE
  Specifies no IP address and no options are served to the specified client ID.

  For example:

  ```
  client 6 10005aa4b9ab NONE
  ```

- Exclude an IP address.

  To exclude one or more IP addresses from the pool of addresses available for lease, use the following statements:

  ```
  client 0 0 192.67.3.123
  client 0 0 192.67.3.222
  ```

  In this case, the hardware type and the client ID are 0. IP addresses 192.67.3.123 and 192.67.3.222 are excluded. You must specify a separate statement for each address you want excluded.

- Exclude a range of IP addresses.

  To exclude a range of IP addresses from the pool of addresses available for lease, specify many `client` statements.

  Each range of excluded addresses should contain no more than 10 addresses. Each excluded address results in a separate `client` statement in the configuration file. To exclude larger numbers of addresses, define subnets that do not include the addresses you want excluded. For example, to exclude addresses 50-75 in subnet 192.67.3.0, specify:

  ```
  inOrder: WIRE1
  subnet 192.67.3.0 255.255.240.0 192.67.3.1-192.67.3.49 label:WIRE1/1
  subnet 192.67.3.0 255.255.240.0 192.67.3.76-192.67.3.100 label:WIRE1/2
  ```

Vendor statement: To provide vendor configuration information to the DHCP clients in your network:

- Define a vendor and assign the appropriate configuration values. Unlike the Class statement, you cannot control the scope of the Vendor statement by its

placement in the file. Use the Vendor statement only at the global level. Vendor statements within Subnet, Class, or Client statements are ignored. Options can be redefined in the vendor class.

- The DHCP client identifies itself to the DHCP server as belonging to a vendor class by sending Option 60 (Class Identifier) with a specific vendor name.
- The DHCP server recognizes that the client has a specific vendor and returns encapsulated Option 43 (Vendor-specific Information), which contains vendor-specific DHCP options and option values.

To define a vendor, use the following statement:

```
vendor vendor_name [hex value]
```

vendor_name
The user-defined label that identifies the vendor. The vendor name is an ASCII string of up to 255 characters (for example, ″IBM″). If the vendor name contains spaces, it must be surrounded by quotes (″).

[hex *value***]**
value must be specified either as an ASCII string or as hexadecimal in the hexadecimal ASCII string construct. For example:

```
hex "01 02 03"
```

For more information, see descriptions of Option 60 (Class-Identifier) in “Specifying DHCP options” on page 520.

The vendor statement can also be specified in the DHCP server configuration file as a vendor statement followed by a pair of braces containing the options particular to this vendor. Within these braces, the usual option value encoding and decoding rules do not apply.

Statements specifying server information:

- Querying in-use address.

  Before the server allocates an IP address, it PINGs the address to make sure that it is not already in use by a host on the network. The server places an in-use address in a special pool and allocates a different address.

  To specify the amount of time a DHCP server holds an in-use address in a special pool before returning the address to the active pool available for assignment, use the following statement:

  ```
  usedIPAddressExpireInterval in_use_time_value
  ```

  in_use_time_value is a decimal integer optionally followed by a space and a unit of time, which can be years, months, weeks, days, hours, minutes, or seconds. The default is 1000 seconds. If you do not specify a unit of time, minutes are assumed.

  Default interval:
  1000 seconds

  Default unit:
  minute

  Minimum:
  30 seconds

  Maximum:
  -1, which is infinity

- Transform canonical addresses.

For 802.3 clients, use the canonical keyword to instruct the DHCP server to transform MAC addresses to canonical (byte starts with least significant bit) form. In most cases, you do not want the DHCP server to transform canonical addresses. MAC addresses of 802.3 clients are normally in canonical format on an 802.3 network. When 802.3 MAC addresses are transmitted across a transparent bridge, the bridge reformats the bits that identify an 802.3 client MAC address to a noncanonical (byte starts with most significant bit) form. When the bridge returns the MAC address to an 802.3 network, the bridge again reformats MAC addresses.

To cause the DHCP server to transform MAC addresses, use the following statement:

`canonical [YES | NO]`

NO prevents the DHCP server from transforming MAC addresses. YES causes the DHCP server to transform MAC addresses. NO is the default value.

- Specify statistics snapshots.

  To specify the number of intervals that expire before the DHCP server takes a snapshot of statistics, use the following statement:

  `statisticSnapshot` *`number_of_intervals`*

  The length of each interval is determined by the `leaseExpireInterval` keyword. For example, a value of 3 collects statistics after a span of three intervals, where each interval has a length specified by the `leaseExpireInterval` keyword. If no value is specified, the server takes a snapshot of statistics at the end of every lease expire interval.

Additional options: To assign additional configuration parameters, use the Option statement. All clients inherit all globally-defined options. A client that is defined within a Subnet statement inherits global options and options that are defined for that address pool. To assign configuration parameters for all clients in a subnet, follow the Subnet statement with Option statements that are surrounded by braces.

A sample DHCP server configuration files: The following is a sample DHCP configuration file:

```
logFileName dhcpsd.log
logFileSize 100
numLogFiles 4
logItem SYSERR
logItem ACNTING
logItem OBJERR
logItem EVENT
logItem PROTERR
logItem WARNING
logItem INFO
logItem TRACE
logItem ACTION
supportBootP yes
supportUnlistedClients NO
bootStrapServer 192.168.1.4
option 211 "nfs"
option 212 "10.1.1.2
option 213 "/usr/lpp/nstation/standard/configs/
option 214 "nfs"
option 67 /usr/lpp/nstation/standard/kernel

leaseTimeDefault 12 HOURS

option 15  mycompany.com

# Addresses 8.67.112.24 through 8.67.112.25 do not inherit
# options defined for 8.67.112.26 through 8.67.112.30
```

```
subnet 192.168.1.00 255.255.255.0 192.168.1.1-192.168.1.100
 {
  option 1 255.255.255.0
  option 3 10.1.1.1

  client 0 0 192.168.1.4
  client 0 0 192.168.1.5
 }
```

Specifying DHCP options: DHCP allows you to specify options, also known as BOOTP vendor extensions, to provide additional configuration information to the client. RFC 2132 defines the options that you can use. Each option is identified by a numeric code.

Architected Options 0 though 127 and Option 255 are reserved for definition by the RFC. The DHCP server and the DHCP client use options in this set. The administrator can modify some architected options. Other options are for exclusive use by the client and server. The administrator cannot or should not configure the following options at the DHCP server:

- 52 (Option Overload)
- 53 (DHCP Message Type)
- 54 (Server Identifier)
- 55 (Parameter Request List)
- 56 (Message)
- 57 (Maximum DHCP Message Size)
- 60 (Class Identifier)

Options 128 through 254 represent options that can be defined by administrators to pass information to the DHCP client to implement site-specific configuration parameters. Additionally, IBM provides a set of IBM-specific options, such as Option 192 (TXT RR).

The format of user-defined options is:

`option` *code value*

code can be any option code 1 through 254. *value* must always be a string. At the server, it can be an ASCII string or a hexadecimal string. At the client, however, it always appears as a hexadecimal string that is passed to the processing program.

The server passes the specified value to the client. You must, however, create a program or command file to process the value. For more information about data formats for the DHCP options, refer to *z/OS Communications Server: IP Configuration Reference*.

Option categories: The DHCP options are divided into the following categories:

- Base Options
- IP Layer Parameters per Host Options
- IP Layer Parameters per Interface Options
- Link Layer Parameters per Interface Options
- TCP Parameter Options
- Application and Service Parameter Options
- DHCP Extensions Options

- Load Balancing Options
- IBM-Specific Options

Hardware types: The following is a list of hardware types you can specify on the Client statement:

Type	Description
0	Unspecified. If you specify a symbolic name for the client ID, specify 0 for the hardware type.
1	Ethernet (100MB)
6	IEEE 802 Networks (which include 802.5 token ring)

Configuring DHCP server as DDNS client proxy

The DDNS proxy A-record update feature supports clients that do one of the following:

- Use DHCP option 81 to communicate their host name information to a DHCP server.
- Send some or all of their host name information in DHCP option 12 and option 15.
- Have their host name information defined to the DHCP server by an administrator.

The proxyARec update feature allows non-Dynamic IP clients (clients which have DHCP client capability but not DDNS capability) to effectively participate in the Dynamic IP system. This alternative is well-suited for network environments where client hosts are not mobile or do not change administrative domains.

New keywords in the DHCP server configuration file instruct the DHCP server how to interact with the DDNS server using A records on behalf of DHCP clients. The new keywords are:

proxyARec
: For more information on enabling DHCP server A record updates, see "Enabling dynamic A record updates".

updateDNSA
: For more information on enabling A record updates, see "Specifying the keyword for A record updates" on page 522.

updateDNSP
: For more information on enabling PTR record updates, see "Specifying the keyword for PTR record updates" on page 524.

Enabling dynamic A record updates: Once the DHCP server reads the configuration file and is instructed how and when to perform a DDNS proxy A record update, the following rules of precedence are used to derive the host name data to be used:

1. As a first priority, option 81 is included in the DHCP client request and indicates that an A record update should be performed at the DDNS server on behalf of the client.
2. Either option 12 or option 15, or both, are included in a DHCP client request.
3. Either option 12 or option 15, or both, are defined in the DHCP server configuration file.

The data used in the DDNS update packet is derived using one or more of the above means until a fully qualified host name data is established. For example, in the case of Microsoft® Windows 3.11+ and Windows 95 clients, only option 12, the Microsoft Windows client computer name, is included in DHCP lease requests. In this case, the administrator can define an option 15, domain name, to use with the host name provided by a client's option 12 to obtain the host's fully qualified name.

The proxy A-record update feature works with any client which is capable of including host name info in option 81 and/or options 12 and 15. Otherwise, the DHCP client host, as identified by its LAN adapter's MAC address, and its host name information can be specified entirely in the DHCP server's configuration file.

Note: If you use DDNS proxy A record support for subnets which include some hosts capable of updating their own DDNS A-records such as IBM OS/2 Warp Dynamic IP clients, you might want to disable those client's own DDNS update feature by commenting out updateDNSA and UpdateDNSTxt keywords in the client's DHCPCD.CFG configuration file. If you do not disable the dynamic A record updates at the clients, the proxy A record updates attempted at the DHCP server for those clients will fail because the DHCP client, rather than the DHCP server, will own the client's entry on the DDNS database.

Specify the proxyARec keyword to enable the DHCP server to dynamically update an A record mapping (host name-to-IP-address) on a DDNS server for clients unwilling or unable to update their A records. The format of the proxyARec keyword is:

```
proxyARec
```

The DHCP server performs updates of the client's A record regardless of the client's client ID value.

The proxyARec keyword is required to be enabled globally or within the subnet, class, or client level. The updateDNSA keyword must also be specified. If the proxyARec keyword is not specified, clients will not have their A records updated.

Specify the proxyARec statement within braces to indicate the information applies only to the subnet, class, or clients that meet the criteria of the preceding conditional statement. To globally assign proxyARec, specify the keyword outside any braces.

Specifying the keyword for A record updates: To specify how the server updates A records for a non-IBM IP client, use keyword:

```
updateDNSA command_string
```

The following is a typical updateDNSA string issued by the DHCP server:

```
updateDNSA "nsupdate -f -h%s -s"d;a;*;a;a;%s;s;%s;3110400;q" -q"
```

This default string is normally adequate. Typical changes to the command string are to modify the value of an expiration time (such as 3110400 seconds) beyond the A record expiration. Assuming proxyARec is specified, this example instructs the DDNS name server to delete all A records for this host name and add a record that maps the host name to the specified IP address for the specified lease time.

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-h Client host name (value is substituted by the DHCP server).

-s Indicates that the command should be run in command-line mode and all subcommands are included within the quotation marks that follow. The following command string is appropriate for most cases. The string contains:

d;a;*; Delete all A (host name-to-IP-address) records for this host.

a;a;%s;
Add an A record using an IP address (%s) provided by the server, where %s indicates string substitution.

s;%s; Send a lease time (%s) (value provided by the server for the IP address), where %s indicates string substitution.

3110400;
The effect of this value is to reserve the host name for an additional time interval after the A record expiration. In this case, an interval of 36 days (3110400 seconds) is added to the A record expiration time. This value works to preserve a user's host name during times when the name-to-address mapping might expire, such as holidays, vacation, or other times of inactivity.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

For more information on client control of A record updates for non-dynamic DHCP clients, see client option 81 in Specifying DHCP Options.

Releasing a client A record: Use the releaseDNSA keyword at a global level as a template which tells the DHCP server how to release a client record, when the client requests release. The template is adequate for normal purposes, but can be modified if necessary.

The keyword is:

```
releaseDNSA string
```

The following is an example of a releaseDNSA string issued:

```
releaseDNSA "nsupdate -f -h%s -s"d;a;%s;s;%s;0;q" -q"
```

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-h%s
Client host name (value is substituted by the DHCP server).

-s Information in the command string releases the DHCP client's A record at the DDNS server. The command string is appropriate for most cases. The string contains:

d;a;%s;
Delete the A (host name-to-IP-address) record for this host.

s;%s; Send a lease time (%s) (value provided by the server for the IP address).

0; An additional time interval added to the normal expiration of the host name beyond the expiration of the A resource record. In this case, the value is 0 because the A resource record is deleted.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

Specifying the keyword for PTR record updates: DHCP servers own the PTR records and can update DDNS servers with a PTR record (resource records that map an IP address to a host name) for each address allocated to a host that identifies itself with DHCP options 12 (host name) and 15 (domain name), or with option 81, Client DHCP-DNS.

To enable DHCP server PTR record updates, use:

```
updateDNSP command_string
```

The following is a typical updateDNSP string issued by the DHCP server:

```
updateDNSP "nsupdate -f -r%s -s"d;ptr;*;a;ptr;%s;s;%s;0;q" -q"
```

This default string is normally adequate. Typical changes to the command string are to modify the name expiration extension time such as 0. This example instructs the DDNS name server to delete all PTR records for this address and add a record that maps the IP address to the new host name for the specified lease time. The updateDNSP command is issued when the DHCP client is assigned an address.

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-r%s
The IP address which is to be mapped to the new host name.

-s The command should be run in command-line mode and all subcommands are included within the quotation marks that follow. The command string is appropriate for most cases. The string contains:

d;ptr;*;
Delete all PTR (IP-address-to-host name) records for this IP address.

a;ptr;%s;
Add a PTR record which maps the IP address to the host name (%s) provided by the server.

s;%s; Send a lease time (%s) (value provided by the server).

0 The effect of this value is to reserve the IP address entry this many seconds beyond when the IPaddress-to-host name mapping expires. In this case, no additional time is added.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

Note: The updateDNSP keyword is equivalent to the updateDNS keyword in earlier releases. The updateDNS keyword continues to be supported.

Releasing a client PTR record: Use the releaseDNSP keyword at a global level as a template which tells the DHCP server how to release the client PTR record, when the client requests release. The template is adequate for normal purposes, but can be modified if necessary. For example, this DHCP server keyword enables immediate release of a PTR record when a mobile client's lease is terminated.

The DHCP server command is:

```
releaseDNSP string
```

The following is an example of a releaseDNSP string:

```
releaseDNSP "nsupdate -f -r%s -s"d;ptr;%s;s;%s;0;q" -q"
```

The command string includes:

nsupdate
The name of the DDNS client program, which updates the DDNS database.

-f The request originates from a DHCP server.

-r%s
The IP address substituted by the DHCP server.

-s Information in this command string releases the DHCP client's PTR (IP-address-to-host-name mapping) record at the DDNS server. The command string is appropriate for most cases. The string contains:

d;ptr;%s;
Delete the PTR record for this address.

s;%s; Send a lease time (%s) (value provided by the server for the IP address).

0 An additional time interval added to the normal expiration of the PTR resource record beyond the expiration of the PTR resource record. In this case, the value is 0 because the record is deleted.

q Quit delimiter for the command string.

-q Nsupdate will process the request in quiet mode.

Defining DHCP proxy authority

Defining DDNS support also includes ensuring the DHCP domain is recognized at the DDNS server and setting up the necessary security for making updates to the DDNS server. To enable A record and PTR record dynamic update support in your DHCP server:

- Ensure that the administrator of the DDNS server configures dynamic reverse domains (*.in-addr.arpa) for all addresses for which the DHCP server will be making dynamic PTR updates.
- Create key file entries in the DDNS.DAT file for each dynamic reverse domain to be updated. If you use DDNS in your network, you must also create a DDNS.DAT file, which contains the security keys used to process IP address and host name updates with a DDNS server.

Defining DDNS key files for the PTR record: The DDNS.DAT file contains the security key pairs to be used for processing IP address-to-host name and host name-to-IP-address updates for the specified addresses which are sent to the corresponding primary DDNS server. The file must contain at least one entry per primary DDNS server.

Run the nsupdate command on the server to create key entries to the /etc/ddns.dat file for DHCP supported clients. The format of the nsupdate command for creating security key pairs for updating PTR records in the DDNS.DAT file is:

```
nsupdate -f -hinverse_ip_address -g -pprimary_ddns_server
```

The command string includes:

-f The request originates from a DHCP server.

-h The inverse IP addresses for which keys are generated.

inverse_ip_address
The format of the inverse_ip_address is the IP address in reverse with in-addr.arpa appended. For example, 9.67.149.111 as an inverse_ip_address entry is 111.149.67.9.in-addr.arpa. You can use a wildcard to expand the scope of the entry. For example, if you want this entry to apply to all hosts whose IP address begins with 9.67, you could specify *.67.9.in-addr.arpa. If you want this entry to apply to all hosts, you could specify *.in-addr.arpa.

-g A key pair should be created for this entry.

-p The primary DDNS server.

primary_ddns_server
The host name or IP address of the primary DDNS server for the specified addresses.

For example:

```
nsupdate -f -g -h *.33.37.9.in-addr.arpa -p ns-updates.company.com
```

The wildcard (*) allows aggregate entries such as *.33.37.9.in-addr.arpa to represent all addresses beginning with the three octets 9.37.33.

The format of the entry in the DDNS.DAT file for a DHCP server is:

```
inverse_ip_address  primary_ddns_server  private_key  public_key
```

The inverse_ip_address and primary_ddns_server are explained above.

The private_key and public_key are used to provide fail-save authentication for updates submitted by this DHCP server. As stored in the DDNS.DAT file, the private key is encrypted.

Defining DDNS key files for the A record: The format of the nsupdate command for creating security key pairs for updating A records in the DDNS.DAT file is:

```
nsupdate -f -g -hfully_qualified_host_name -pprimary_ddns_server
```

The command string includes:

-f The request originates from a DHCP server.

-h The host names for which keys are generated.

fully_qualified_host_name
The format of the fully_qualified_host_name is the host name followed by the fully qualified domain name or a wildcard to allow aggregate entries.

-g A key pair should be created for this entry.

-p The primary DDNS server.

primary_ddns_server
The host name or IP address of the primary DDNS server for the specified addresses.

For example:

```
nsupdate -f -g -h *.city.company.com -p ns-updates.company.com
```

You can specify a specific host, such as myhost, or use a wildcard to allow aggregate entries such as *.city.company.com to represent all hosts in the zone.

Configuring the BINL server

The BINL server (binlsd) is similar to the DHCP server and is started and configured similarly. Port 4011 must be reserved for the BINL server. To reserve port 4011 for binlsd, add the following line to the PORT statement of the TCPIP profile:

```
4011 UDP BINLSD
```

Also, if the system is running in a multiple stack (common INET) environment, the INADDRANYPORT range cannot include 4011.

Following are the command line options that can be used with binlsd:

Option	Description
-?	Display the help message.
-b	Display the program banner.
-q	Execute in quiet mode.
-v	Execute in verbose mode.
-f	Override default configuration file location.

The BINL configuration file, dhcpsd.cfg by default, is searched for first in the current working directory. If not found in the current working directory, the /etc directory is searched. Following is a sample BINL configuration file shipped as /usr/lpp/tcpip/samples/binlsd:

```
#########################################################################
#
#   binlsd.cfg -- BINL (Image Server) Configuration File
#
#   This file contains the directives that can be specified by the
#   server's administrator to configure the server and enforce
#   policies. This file is only a sample.  The finished file must be
#   placed in the directory specified by the ETC environment variable.
#
#   Do not put any long line without spaces in this file.
#
#   A line starting with a '#' character is a comment and is ignored.
#   A '#' on a line which is not part of a quoted string indicates
#   that anything to the right of this character is a comment and should
#   be ignored.
#
#   A continuation character of '\' is supported.  It must be
#   the last non-whitespace character on the line prior to any comments.
#
#   The directives are specified in the form of
#   <keyword><value1>...<valueN>.
#
#   Here is a partial list of keywords whose value can be specified
#   in this file:
#
#   Keyword             Effect
#   -------------       ----------------------------------------------------------------
#   sa                  Specifies the system architecture.
#   nit                 Specifies the network interface type.
```

```
#   lsa1nic             Specifies the lsa1 nic type.
#   tftp                Specifies the address of the tftpd server.
#   bpname              Specifies the install filename given to clients.
#   numLogFiles         The number of log files desired.
#   logFileSize         The size of log files in kilobytes.
#   logFileName         The name of the most recent log file.
#   logItem             An item to be logged.
#   servername          The LCM server name for LSA-1 clients. Not used for LSA-2 clients.
#   serverdomainname    The LCM domain name for LSA-1 clients. Not used for LSA-2 clients.
#
#   client              Definition of a set of options for a specific client
#                       or a definition of a client not to be serviced.
#
#   pxevendor           Configuration delimiter to indicate pxe options to follow.
#
#   pxeoption           A pxe configuration option value to pass to clients.
#
#   The scope of a keyword is limited by a pair of curly brackets ({, })
#   within which the keyword is located.  If a keyword is located outside
#   of any pair of curly brackets, its scope is applicable to all the
#   clients served by this server.  The curly brackets must appear alone on
#   a line.
#
#  Log files.  This set of parameters specifies the log files that will be
#  maintained by this server.  Each parameter is identified by a keyword
#  and followed by its value.
#
#  Keyword      Value           Definition
#  --------     ------------    ----------------------------------------
#  numLogFiles  0 to 99         number of log files.  If 0 is specified,
#                               no log file will be maintained and no log
#                               message is displayed anywhere.  When a log
#                               file reaches maximum size, a new log file
#                               is created, until the maximum number of
#                               log files have been created.  Only the most
#                               recent n log files are kept/
#
#  logFileSize  in K bytes      maximum size of a log file.  When the size
#                               of the most recent log file reaches this
#                               value, it is renamed and a new log file is
#                               created.
#
#  logFileName  file path       name of the most recent log file.  Less
#                               recent log files have the number 1 to
#                               n-1 appended to their names; the larger
#                               the number, the less recent the file.
#
#  logItem                      An item that will be logged.
#               SYSERR          System error, at the interface to the platform.
#               OBJERR          Object error, in between objects in the process.
#               PROTERR         Protocol error, between client and server.
#               WARNING         Warning, worth of attention from the user.
#               EVENT           Event occurred to the process.
#               ACTION          Action taken by the process.
#               INFO            Information that might be useful.
#               ACNTING         Accounting information on clients served.
#               TRACE           Code flow, for debugging.
#
#

#  servername   <servername>
#                               If present this name is sent to LSA-1 clients in the r
#                               offer packet.  If this option in not present the server
#                               will send its name to the client as the LCM server.
#  serverdomainname <domainname> <workgroup|domain>
#                               Specifies the domain name of the LCM server that is sent
#                               to LSA-1 clients in the offer packet.
#
#                               <domainname> the name of the domain
#
#                               The last parameter can be the string "workgroup" to
#                               indicate that <domainname> is a workgroup or the
#                               string "domain" to indicate that <domainname>
#                               is a domain.

#  client <id_type><id_value><exclude|value>
```

```
#                               Definition of a client record.
#
#                               <id_type> is one of the hardware types defined
#                               in RFC 1340 (e.g. 1 for 10 megabit Ethernet,
#                               6 for 802.5 Token Ring.)  The type may be
#                               0, in which case the hardware type is not
#                               specified and the
#                               id_value may be a string of any format.
#
#                               <id_value> is a character string if the type
#                               is 0. Typically, this would be a domain name.
#                               For a non-zero <id_type>, the <id_value> is
#                               a hexadecimal string representing
#                               the hardware address of the client.
#
#                               The last parameter can be blank to indicate that
#                               the client matching <id_type> and <id_value>
#                               should get the specific parameters defined in
#                               its scope.
#
#                               The last parameter can be the string "exclude" to
#                               indicate that the client matching
#                               <id_type> and <id_value> should
#                               not be serviced by this server.
#
#       The client statement may be immediately followed
#       by a pair of curly brackets, in which the options
#       particular to this client can be specified.
#
#       Note: All clients inherit all globally defined options.
#
#
#  Pxeoption.  This keyword identifies an option statement.  The options assigned
#  are defined in the "Wired for Management Baseline Version 1.1" specification
#  authored by the Intel Corporation.
#
#  An option is specified by the "pxeoption" keyword followed by the option code
#  of this option and its data field, in a single line.  One or more options
#  may be specified.
#
#  The scope within which an option applies is delimited by a pair of curly
#  brackets ({, }) surrounding the option statement.
#
#  If two or more options with the same option code are specified, the
#  one with the most specific scope is used.  This allows, for example,
#  an option specified at the sa and nit scope to override that same option
#  specified at the global scope.  If two or more options
#  with the same option code are specified within the same scope,
#  the first one read by the server will be the one used, (subject to
#  its being overridden by the same option in a more specific scope).
#
#  All options specified will be sent to the client encapsulated in DHCP option 43.
#

#  Sa.  This parameter specifies the system architecture.  At the global level sa is
#  considered to be "unspecified" and any sa which is received will match if a more
#  specific sa is NOT explicitly configured.  A specific sa may be configured and
#  values for boot path name, TFTP server, and options maybe scoped under this
#  specific specification.
#
#  NIT. This parameter specifies the network interface type. Works in a similar manner
#  as the system architecture (Sa.)  It can be configured to more specifically qualify
#  a SA (e.g sa 0 nit 1) or if configured alone will be considered to more specifically
#  qualify the unspecified SA.
#
#  LSA1NIC.  This parameter specifies the network interface type for LSA-1 clients.
#  Works in a similar manner as the system architecture (Sa.) and network interface
#  type (NIT.) Specific lsa-1 nic types to provide values for boot path name and
#  TFTP server.  No additional options are sent to the LSA-1 client.
#
#
#  TFTP.   This parameter specifies the address of the tftp server that contains
#  the install image.
#
#  Keyword             Value                Definition
```

```
#  -----------   ----------------------    --------------------------
#  tftp          [ipaddress|hostname]      If "ipaddress" actual address of image server.
#
#                                          If "hostname" dns lookup is done to resolve
#                                              ipaddress of image server.
#
#
#  bpname.  The fully qualified pathname of the install image file.
#
#  Keyword           Value
#  -----------     -------------------
#  bpname         [filename]
#
#
################################################################################

# The remaining portion of this file is an sample configuration file.
# Comments are added to assist in understanding the configuration file.
# Further information and detail is found in the online user documentation.

# Setup of the log file information.  This includes the size and name of the
# logfile along with number of logfiles maintained and type of information that
# will be logged.
numLogFiles     10
logFileSize     1000
logFileName     binlsd.log
logItem         SYSERR
logItem         OBJERR
logItem         PROTERR
logItem         WARNING
logItem         EVENT
logItem         ACTION
logItem         INFO
logItem         ACNTING
logItem         TRACE

# default TFTP server
tftp   9.33.44.55

# Fully qualified boot Image pathname
bpname c:\tftp\lccm\lccm.1

# Global Options
pxevendor
{
  pxeoption 2 555
  pxeoption 3 544
  pxeoption 4 5
  pxeoption 5 5
}
#Excluded client list
client 1 080aab4567 exclude
client 1 000abc45d7 exclude
client 6 080aab4899 exclude

sa 0 nit 1
{
  tftp 9.180.71.79
  bpname c:\tftp\lccm\lccm.1

  #Options specific to this sa and nit values
  pxevendor
  {
    pxeoption 1 224.1.1.1
    pxeoption 2 1758
    pxeoption 3 1759
    pxeoption 4 1
    pxeoption 5 2
  }
}

nit 2
{
  tftp 9.180.71.79
  bpname c:\tftp\lccm\lccm.1
```

```
  #Client excluded from service if it falls into this category.
  client 6 08aa343bf3 exclude

  # Special configuration for client following into this nit2 type
  client 1 12345abcd34
  {
    tftp 9.37.56.2
    bpname D:\intel\nit2\nt40s
  }
}

lsa1nic 53
{
  tftp 9.180.71.79
  bpname  d:\lcm\system\images\53.X

  # Special configuration for client following into this nic type
  client 1 12345abcd34
  {
    tftp 9.37.56.2
    bpname D:\intel\nic2\nt40s
  }

  # Deny this client LSA1 service
  client 6 08005aab6abc4 exclude

}

#
# end of binlsd.cfg
#
```

Configuring a DDNS server (BIND 4.9.3 only)

There is no explicit configuration utility for the DDNS server as there is for the DHCP server. You can either create new DDNS server configuration files, or you can migrate an existing DNS configuration to dynamic DNS server configuration files.

There are three ways to use the DDNS server:

- Static DDNS server
- Dynamic secured DDNS server
- Dynamic presecured DDNS server

When used as a static DDNS server, there is nothing you have to do but use your existing DNS configuration files with the DDNS server. It will then work exactly the same way as the previous DNS server.

When used in dynamic secured mode, the DDNS server will allow clients to update their resource records dynamically using encryption keys that have been created by the clients themselves.

When used in dynamic presecured mode, the DDNS server will only allow those clients to update their records to which an encryption key has been provided that has been generated by the administrator.

Creating a new DDNS server configuration: The files required for a minimum configuration are:

- The nameserver boot file contains the path and file names for any other configuration files. The default for this file is /etc/named.boot. It will be examined by the DDNS server at startup.

- The nameserver domain file contains information about the zones for which this server will be authoritative, and all mappings from names to IP addresses (ordinary or forward name resolution). The file name is defined in the nameserver boot file.
- The nameserver reverse file contains information about the mappings from IP addresses to names (inverse or reverse name resolution). The file name is defined in the nameserver boot file.

Use the following steps to create DDNS server configuration files from scratch:

1. Create the DDNS configuration files.

   A nameserver boot file might look as follows:

```
;
; boot file for name server configuration.
;
directory /test/dns/zones/
;
; type       domain                      source file or host
;
primary  test.itsc.raleigh.ibm.com        test.data       dynamic secured
;
primary  200.200.200.in-addr.arpa        db.200.200.200  dynamic secured
;
```

   On the primary statements, you can specify if you want to use the DDNS server in dynamic secured or in dynamic presecured mode by using either the *dynamic secured* or the *dynamic presecured* keywords. A nameserver domain file might look as follows:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
@   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
             ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
    IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
localhost    IN A     127.0.0.1
;
ns-updates   IN A     200.200.200.2
martin       IN CNAME ns-updates
;
BPClient     IN A     200.200.200.14
;
```

   A nameserver reverse file might look as follows:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
200.200.200.in-addr.arpa   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
                                    ns-updates.test.itsc.raleigh.ibm.com. (
        95111601 ; Serial number for this data (yymmdd##)
        86400    ; Refresh value for secondary name servers
        300      ; Retry value for secondary name servers
        86400    ; Expire value for secondary name servers
```

```
        3600     ; Minimum TTL value
         300   ) ; dynamic update increment time
200.200.200.in-addr.arpa.  IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
;
; Addresses for the canonical names
;
2          IN PTR martin.test.itsc.raleigh.ibm.com.
14         IN PTR BPClient.test.itsc.raleigh.ibm.com.
;
```

2. After you have created the files, use the NSUPDATE -g command to create the encryption key pairs for the zone resource records in the domain and reverse files.

   After the administrator has copied the public key into the files, they might look as follows:

   - Domain Name file:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
@   IN  KEY     80 0 1 AQPOzUYWvAUyZhYxogDcrtxOZOH33V31Tmrs1Db1WYiyI4Y
                       7Mmoz6Vm3XY/QTMHOyeHcVAMKmuba+rW4/+IkMeP3
@   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
             ns-updates.test.itsc.raleigh.ibm.com. (
         95111601 ; Serial number for this data (yymmdd##)
         86400    ; Refresh value for secondary name servers
         300      ; Retry value for secondary name servers
         86400    ; Expire value for secondary name servers
         3600     ; Minimum TTL value
          300   ) ; dynamic update increment time
    IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
localhost    IN A     127.0.0.1
;
ns-updates   IN A     200.200.200.2
martin       IN CNAME ns-updates
;
BPClient     IN A     200.200.200.14
;
```

   - Reverse Name file:

```
;
;*******************************
;*  Start of Authority Records  *
;*******************************
;
200.200.200.in-addr.arpa   IN  KEY     80 0 1 AQPR+3ObXCgcjmBfKSnN4fD6v
                                              VH/AUIwincGNeD1MAuz2BTQSQ
                                              /bJkXLA3nxfV+HxKfxWptkRck
                                              wzxEk1DD3DSB
200.200.200.in-addr.arpa   IN  SOA  ns-updates.test.itsc.raleigh.ibm.com.
                                    ns-updates.test.itsc.raleigh.ibm.com. (
         95111601 ; Serial number for this data (yymmdd##)
         86400    ; Refresh value for secondary name servers
         300      ; Retry value for secondary name servers
         86400    ; Expire value for secondary name servers
         3600     ; Minimum TTL value
          300   ) ; dynamic update increment time
200.200.200.in-addr.arpa.  IN  NS   ns-updates.test.itsc.raleigh.ibm.com.
;
;
; Addresses for the canonical names
```

```
;
2          IN PTR martin.test.itsc.raleigh.ibm.com.
14         IN PTR BPClient.test.itsc.raleigh.ibm.com.
;
```

The NSUPDATE -g command will also create the DDNS.DAT file that contains the private encryption keys to sign any updates to the zone resource records in the domain and reverse files. This is shown in the following example:

```
test.itsc.raleigh.ibm.com ns-updates.test.itsc.raleigh.ibm.com
Pb7bySIfzXcW...

200.200.200.in-addr.arpa ns-updates.test.itsc.raleigh.ibm.com
KlexSRMP/q/k...
```

3. Start the DDNS server.
4. If you have a DHCP server configured for DDNS updates, you need to add an entry into the DDNS.DAT file using NSUPDATE -g that represents a wildcard entry for the zones that DHCP will update (*.test.itsc.raleigh.ibm.com).

Note: KEY and SIG resource records, as well as encryption keys, always use a single line. The examples were indented for illustration purposes only.

Configuring for presecured mode operation: IBM Open Edition's Dynamic DNS server supports two modes of securing updates to a dynamic DNS zone. In the default mode, called *dynamic secured*, any host that complies with the DDNS protocol can create resource records in a zone declared as dynamic. After created, these records can only be updated by the administrator or by the host that created them.

In presecured mode, only hosts that have been preauthorized by a DDNS administrator may create resources in a particular dynamic zone. After created, these records can only be updated by the administrator or by the host that has been preauthorized.

Functionally, the difference in these modes is whether or not the DDNS server will allow the creation of a KEY RR or whether it must already have a KEY RR defined for a resource in the zone. In the case of presecured mode, the KEY RR must be already defined in the zone before an update is accepted. Specifically, the administrator must enter the KEY RR data in the domain file for each client that will be making updates.

To configure a particular DNS dynamic zone for presecured mode operation, you must:

1. Specify **dynamic presecured** on the primary zone statement in the DDNS server boot file and create the corresponding domain file.
2. Run the nsupdate -g command to create an /etc/ddns.dat with the zone key information. Create a zone KEY RR.
3. Start the DDNS server by entering **named** at a z/OS UNIX command prompt.
4. For each client host:
   a. Use **nsupdate** with the -g parameter, which will generate a key for the host and save it in /etc/ddns.dat.
   b. Use **nsupdate** with the -a parameter to dynamically register and save a host's KEY RR in the DDNS server domain file.
   c. Manually extract the /etc/ddns.dat file key entry for the host into a separate DDNS.DAT file and distribute it to the end user for installation into their /etc subdirectory.

The following example is a scenario demonstrating an administrator using **nsupdate** to preregister an end user in a **dynamic presecured** domain. Administrator input is highlighted in bold:

```
#nsupdate -g -h newuser.dynozone.sandbox -p netadmin.dynozone.sandbox
--- NSUPDATE Utility ---
---
Key Gen ...... succeeded ...


#nsupdate -a -h newuser.dynozone.sandbox -p netadmin.dynozone.sandbox
--- NSUPDATE Utility ---

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> a
..rrtype (A,PTR,CNAME,MX,KEY,HINFO): key
DDNSUpdate_KEY (Add Flags 0000 Protocol 0 Algid 1
      Keylen 64 Key[0-15]:       AQO/3Ah6986cXDhR  ...        succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> s
..sig Expiration (secs from now, ENTER for 3600):
..sig KEY pad (ENTER for default of 3110400):
DDNSSignUpdate ...succeeded
DDNSFinalizeUpdate ...succeeded
DDNSSendUpdate ...succeeded

Enter Action (Add,Delete,Exists,New,TTL,Send,Quit)
> q

#
```

DNS-related RFCs

The following RFCs contain basic information about the DNS:

974 *Mail Routing and the Domain System*, C. Partridge

1033 *Domain Administrators Operations Guide*, M. Lottor

1034 *Domain Names—Concepts and Facilities*, P.V. Mockapetris

1035 *Domain Names—Implementation and Specification*, P.V. Mockapetris

Proposed standards

2181 *Clarifications to the DNS Specification*, R. Bush Elz

2308 *Negative Caching of DNS Queries*, M. Andrews

1995 *Incremental Zone Transfer in DNS*, M. Ohta

1996 *A Mechanism for Prompt Notification of Zone Changes*, P. Vixie

2136 *Dynamic Updates in the Domain Name System*, P. Vixie, S. Thomson, Y. Rekhter, and J. Bound

2845 *Secret Key Transaction Authentication for DNS (TSIG)*, P. Vixie, O. Gudmundsson, D. Eastlake, 3rd, and B. Wellington

Proposed standards still under development

1886 *DNS Extensions to support IP version 6*, S. Thomson and C. Huitema

2065 *Domain Name System Security Extensions*, D. Eastlake, 3rd and C. Kaufman

2137 *Secure Domain Name System Dynamic Update*, D. Eastlake, 3rd

Other important RFCs about DNS implementation

1535 *A Security Problem and Proposed Correction With Widely Deployed DNS Software*, E. Gavron

1536 *Common DNS Implementation Errors and SUggested Fixes*, A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller

1982 *Serial Number Arithmetic*, R. Elz and R. Bush

Resource record types

1183 *New DNS RR Definitions*, C.F. Everhart, L. A. Mamakos, R. Ullmann, and P. Mockapetris

1706 *DNS NSAP Resource Records*, B. Manning and R. Colella

2168 *Resolution of Uniform Resource Identifiers using the Domain Name System*, R. Daniel and M. Mealling

1876 *A Means for Expressing Location Information in the Domain Name System*, C. Davis, P. Vixie, T., and I. Dickinson

2052 *A DNS RR for Specifying the Location of Services*, A. Gulbrandsen and P. Vixie

2163 *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping*, A. Allocchio

2230 *Key Exchange Delegation Record for the DNS*, R. Atkinson

DNS and the Internet

1101 *DNS Encoding of Network Names and Other Types*, P. V. Mockapetris

1123 *Requirements for Internet Hosts - Application and Support* Braden

1591 *Domain Name System Structure and Delegation*, J. Postel

2317 *Classless IN-ADDR.ARPA Delegation*, H. Eidnes, G. de Groot, and P. Vixie

DNS operations

1537 *Common DNS Data File Configuration Errors*, P. Beertema

1912 *Common DNS Operational and Configuration Errors* D. Barr

2010 *Operational Criteria for Root Name Servers*, B. Manning and P. Vixie

2219 *Use of DNS Aliases for Network Services*, M. Hamilton and R. Wright

Other DNS-related RFCs

1464 *Using the Domain Name System To Store Arbitrary String Attributes,* R. Rosenbaum

1713 *Tools for DNS Debugging* A. Romao

1794 *DNS Support for Load Balancing*, T. Brisco

2240 *A Legal Basis for Domain Name Allocation*, O. Vaughan

2345 *Domain Names and Company Name Retrieval*, J. Klensin, T. Wolf, and G. Oglesby

2352 *A Convention For Using Legal Names as Domain Names*, O. Vaughan

Chapter 11. Policy-Based Networking

The role of policy

Businesses typically define goals for network behavior in human terms, for example using Service Level Agreements (SLAs) or defining the behavior of Intrusion Detection Services (IDS). Network implementations provide a wide variety of controls for priority treatment of traffic, bandwidth management, and control of network behavior. The link between the high level business goals and network implementations is defined as Policy Based Networking. The implementation of SLAs, IDSs and other network controls on network hosts and routers is provided by *policies*. Policies are an administrative means to define controls for a network, in order to achieve the Quality of Service (QoS) levels promised by SLAs, or to implement IDS or resource balancing decisions. To be effective, especially for QoS, consistent mechanisms need to be used throughout the network to classify and differentiate traffic, and to provide a consistent implementation of policy decisions. To achieve this, policies are usually defined in a central policy repository, accessed by all hosts and routers that need to make policy decisions (*Policy Decision Point* or PDP), or implement such decisions (*Policy Enforcement Point* or PEP).

It is important to be able to monitor the performance of policies as implemented by the network. This leads to the need to define service level management information kept by the PDP and PEP, which can then be analyzed for a variety of network services, from traffic trend analysis to network capacity planning and dynamic QoS level tuning.

Together, service differentiation, policies, and service level performance monitoring form an integral part of the SLA function that is becoming more and more important to network administrators as a means of controlling network traffic flows.

Policy components overview

Policy Agent

Network administrators can use the z/OS CS Policy Agent to define policies for their users. Superuser authority is required.

The policies supported by the Policy Agent can be used for any of the following purposes:

- Quality of Service (Refer toChapter 12, “Quality of Service (QoS)” on page 565)
- Intrusion Detection Services (Refer to Chapter 13, “Intrusion Detection Services (IDS)” on page 595)

The common Policy Agent functions are further described in this chapter.

The Policy Agent supports QoS functions other than reading and installing policies, such as Sysplex Distributor policy performance monitoring, and mapping Type of Service (ToS) byte values to outbound interface and virtual LAN (VLAN) user priorities. The QoS specific Policy Agent functions are further described in “QoS specific Policy Agent functions” on page 567.

The Policy Agent runs in the z/OS environment and reads policy definitions from a local configuration file and/or a central repository that uses the Lightweight Directory

Access Protocol (LDAP). The Policy Agent also installs policies in one or more z/OS CS stacks. It can be used to replace existing policies or update them as necessary.

Notes:

1. Policies are supported by the stack as an end host system only. When the stack is routing packets, policies are not accessed or applied.
2. Policies defined on an LDAP server use the configuration files and mechanisms provided by the LDAP server product. The definition of the elements of policies is known as the *schema*. z/OS CS provides the schema definition for policies that may be defined on an LDAP server in a set of sample files. The sample files are provided in both LDAP protocol version 2 and version 3 format (see "Policy sample files" on page 541 for the names of these samples). These sample files must be installed on the LDAP server as the schema definition. Policy Agent uses the z/OS SecureWay Security Server LDAP Client library to communicate with an LDAP server. Refer to *z/OS Security Server LDAP Server Administration and Use* for more information about LDAP and *z/OS Security Server LDAP Client Programming* for more information about the z/OS CS LDAP Client support.
3. A copy of the policy schema definition files that define the policy schema for LDAP protocol version 2 are also available in the Appendix of the *z/OS Communications Server: IP Configuration Reference*. Different files are used to define the schema in LDAP protocol version 3 syntax, but are not shown because the protocol version 2 syntax is easier to read and also because the schema definitions are essentially the same.

RSVP Agent

The z/OS CS RSVP Agent provides Integrated Services functions, such as communicating with RSVP agents on other hosts or routers and reserving resources on certain types of outbound interfaces. The RSVP Agent queries the Policy Agent for policies that relate to RSVP processing.

SNMP SLA subagent

The z/OS CS SLA subagent allows network administrators to retrieve data and determine if the current set of SLA policy definitions are performing as needed or if adjustments need to be made. The SLA subagent supports the Service Level Agreement Performance Monitor (SLAPM) MIB. Refer to RFC 2758 for more information about the SLAPM MIB.

Intrusion Detection Services

Intrusion Detection Services (IDS) support is available to detect and report on network intrusion events. The Traffic Regulation Management (TRM) support provided in V2R10 has been extended and incorporated into the IDS support. IDS policy regulates the types of events to report and provides the definition of several types of events. IDS policy may be defined for scans, attacks and traffic regulation for both TCP and UDP ports.

Notes:

1. IDS policies may only be defined on an LDAP server.
2. Policy Scope TR policies found in a Policy Agent configuration file are compatibly transformed into IDS TR TCP policies by Policy Agent.
3. pasearch will display the transformed policy.

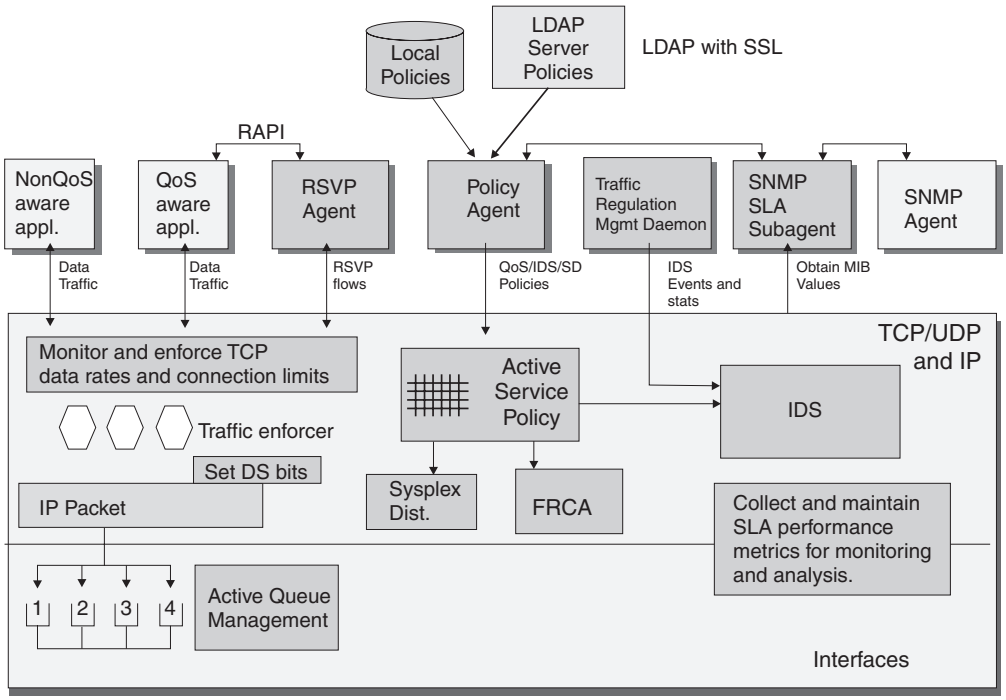


Figure 61. Policy components in z/OS CS

Policy sample files

A set of sample files is shipped with z/OS CS that provides several functions. The first sample file provides an example of policy definitions in a Policy Agent configuration file.

/usr/lpp/tcpip/samples/pagent.conf
: This file contains overall policy definition rules, syntax and semantics for defining policies in a configuration file, and examples of such policy definitions.

The following set of sample files provides an example of policy definitions in LDAP. For more information on using these sample files, see "Using the sample LDAP objects" on page 556.

/usr/lpp/tcpip/samples/pagent.ldif
: This file contains the top level directory structure for the set of sample QoS and IDS policies.

/usr/lpp/tcpip/samples/pagent_starter_QOS.ldif
: This file contains the starter set sample of LDAP definitions of QoS objects. This file requires the directory structure defined in sample file pagent.ldif.

/usr/lpp/tcpip/samples/pagent_starter_IDS.ldif
: This file contains the starter set sample of LDAP definitions of IDS objects. This file requires the directory structure defined in sample file pagent.ldif.

/usr/lpp/tcpip/samples/pagent_advanced_QOS.ldif
: This file contains the advanced set sample of LDAP definitions of QoS objects. This file requires the objects defined in the QoS starter set sample file pagent_starter_QOS.ldif and the directory structure defined in sample file pagent.ldif.

/usr/lpp/tcpip/samples/pagent_advanced_IDS.ldif
This file contains the advanced set sample of LDAP definitions of IDS objects. This file requires the objects defined in the IDS starter set sample file pagent_starter_IDS.ldif and the directory structure defined in sample file pagent.ldif.

The next set of samples is the definition of the schemas in LDAP protocol version 2 format. They must be installed in the LDAPv2 server's configuration as included files. Refer to "Installing the schema definition on the LDAP server" on page 555 for more information.

/usr/lpp/tcpip/samples/pagent_oc.conf
This file contains the schema object class definitions.

/usr/lpp/tcpip/samples/pagent_at.conf
This file contains the schema attribute definitions.

The next set of samples is the definition of the schemas in LDAP protocol version 3 format. They must be installed on the LDAPv3 server in the proper order as an object in the server's database, rather than as configuration information. This process is known as schema publication. Refer to RFCs 1804 and 2251. The files need to be specified on ldapmodify commands to modify the cn:schema entry in the server's database, in the order as specified in the list below. Refer to "Installing the schema definition on the LDAP server" on page 555 for more information.

/usr/lpp/tcpip/samples/pagent_schema.ldif
This file contains the schema version 2 core and QoS schema object class and attribute definitions.

/usr/lpp/tcpip/samples/pagent_v3schema.ldif
This file contains the schema version 3 additions to the schema version 2 core and QoS schemas.

/usr/lpp/tcpip/samples/pagent_schema_updates.ldif
This file contains changes to the schema version 2 core and QoS schema definitions in support of schema version 3.

/usr/lpp/tcpip/samples/pagent_idsschema.ldif
This file contains the schema version 3 IDS schema definitions.

The last set of samples contain the text of draft documents used to develop the Version 3 schema.

/usr/lpp/tcpip/samples/pagent_pcim.txt
This file contains the draft version of the proposed Policy Core Information Model (PCIM) used as the basis for the support in this release. PCIM is described by RFC 3060 but this file is an earlier draft level.

/usr/lpp/tcpip/samples/pagent_core.txt
This file contains the draft version of the proposed Policy Core LDAP Schema Internet Draft used in z/OS V1R2 and later releases.

Note: This file is provided as-is and there are some differences between the draft and the implementation. The intent of this file is to provide background information on the level of the supported schema.

/usr/lpp/tcpip/samples/pagent_cond.txt
This file contains the draft version of the proposed Policy Conditions Internet Draft used in z/OS V1R2 and later releases.

Note: This file is provided as-is and there are some differences between the draft and the implementation. The intent of this file is to provide background information on the level of the supported schema.

Note: This documentation refers to Versions 1, 2 and 3 when defining policies. Version 1 refers to policy definitions used prior to OS/390 V2R10. Version 2 refers to policy definitions used for OS/390 V2R10. Version 3 refers to policy definitions used for z/OS V1R2 and later releases. Version 1 provides only a subset of the functionality of later versions. When defining policies, refer to *z/OS Communications Server: IP Configuration Reference* for more information about V1, V2 and V3 statements.

Policy object model overview

Policies consist of several related objects. The main object is the *policy rule*. A policy rule object refers to one or more *policy condition*, *policy action*, or *policy time period condition* objects, and also contains information on how these objects are to be used. Policy time period objects are used to determine when a given policy rule is active. Active policy objects are related in a way that is analogous to an 'IF' statement in a program. For example:

```
IF condition THEN action
```

In other words, when the set of conditions referred to by a policy rule are TRUE, then the policy actions associated with the policy rule are executed.

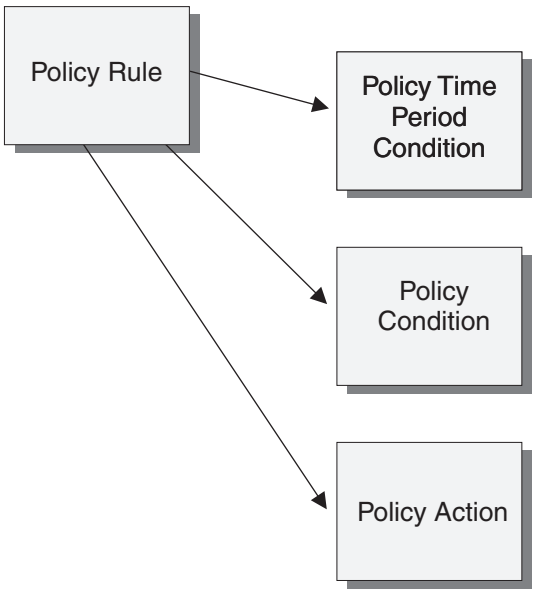


Figure 62. Basic policy objects

Policy rules can refer to one or more policy conditions. A policy rule with a single policy condition is known as a *simple* rule, while one with more conditions is known as a *complex* rule. Complex policy rules can have their conditions evaluated according to one of two different methods. The first is Disjunctive Normal Form (DNF), which means an ORed set of ANDed conditions. The second is Conjunctive Normal Form (CNF), which means an ANDed set of ORed conditions. In order to accomplish these evaluations, individual policy conditions are assigned an arbitrary group number, and also an indication of whether or not the condition is negated. For example, consider the following set of conditions for a policy rule:

```
C1:  Group Number = 1, Condition Negated = FALSE
C2:  Group Number = 1, Condition Negated = TRUE
C3:  Group Number = 1, Condition Negated = FALSE
C4:  Group Number = 2, Condition Negated = FALSE
C5:  Group Number = 2, Condition Negated = FALSE
```

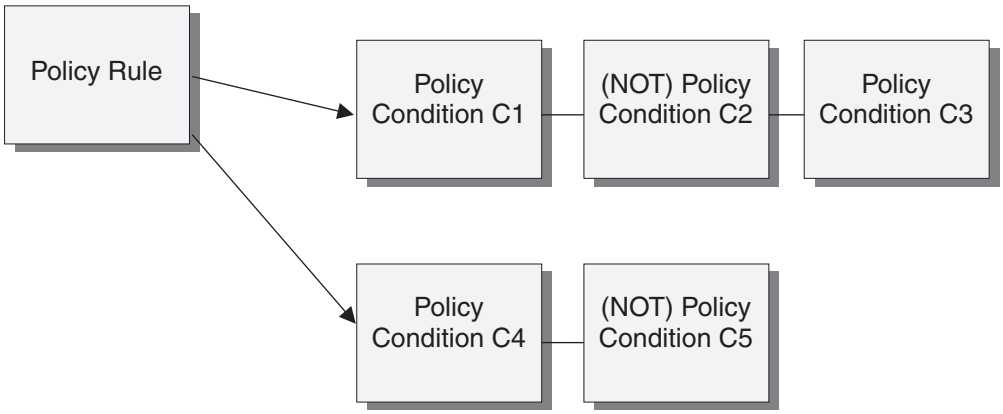


Figure 63. Complex policy conditions

If the conditions are to be evaluated using DNF, then the overall condition for the policy rule is:

```
(C1 AND (NOT C2) AND C3) OR (C4 AND C5)
```

On the other hand, if CNF is used to evaluate the conditions, then the overall condition for the policy rule is:

```
(C1 OR (NOT C2) OR C3) AND (C4 OR C5)
```

Policy actions specify actions to take when the set of conditions for a policy rule evaluate to TRUE. Although the policy model allows multiple actions for a policy rule, in practical terms a single action is all that usually makes sense for the types of policies supported by the Policy Agent.

Policy conditions and actions can either be specific to a single rule, or be reusable among several policy rules. To allow either type of conditions and actions, and to specify related information such as condition group number and negation indicator, several other policy objects are required. First are *policy condition association* and *policy action association* objects. These objects contain condition and action related attributes, respectively, and may directly contain policy conditions and actions (rule-specific).

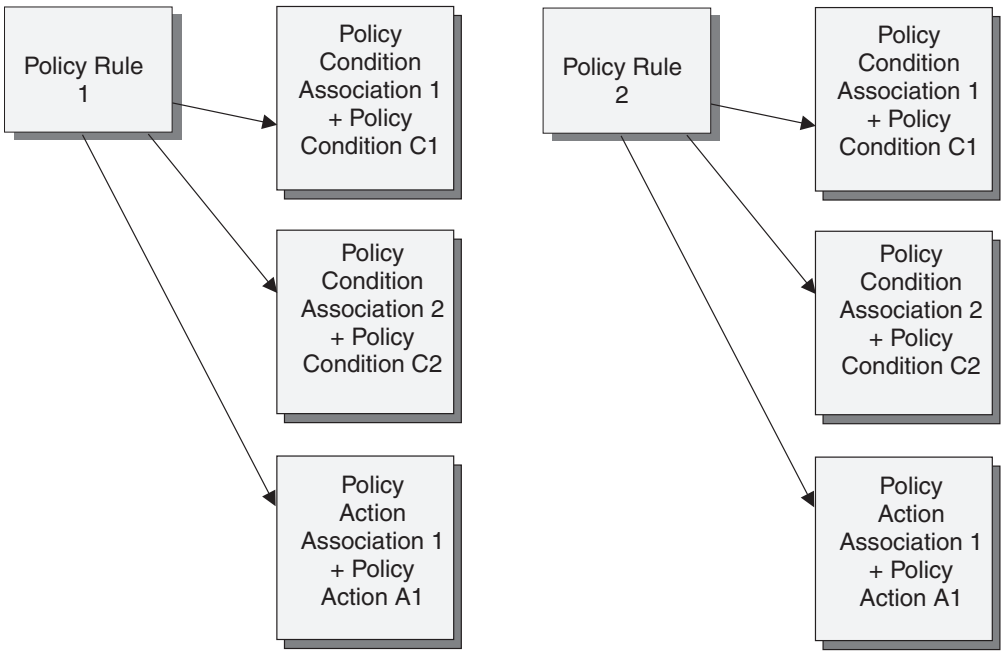


Figure 64. Rule-specific conditions and actions

The policy association objects alternatively may refer to conditions and actions (reusable). *Policy condition instance* and *policy action instance* objects are used to represent reusable policy conditions and actions, respectively.

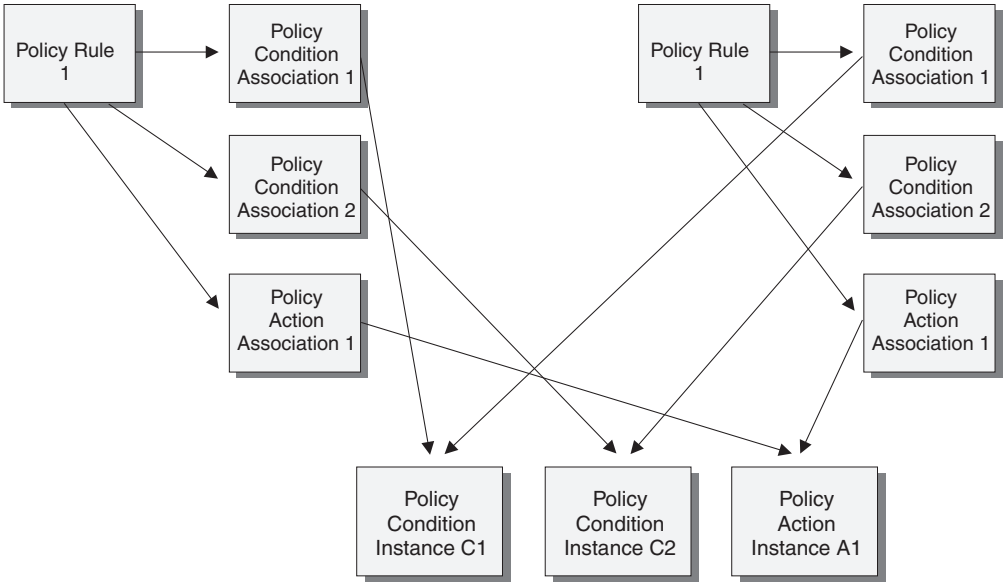


Figure 65. Reusable conditions and actions

Primarily for administrative grouping of policy rules, the *policy group* object is used. Policy groups can refer either to policy rules or to policy groups. This allows related policy rules to be grouped together, and also allows policy groups to be grouped to any needed level of nesting.

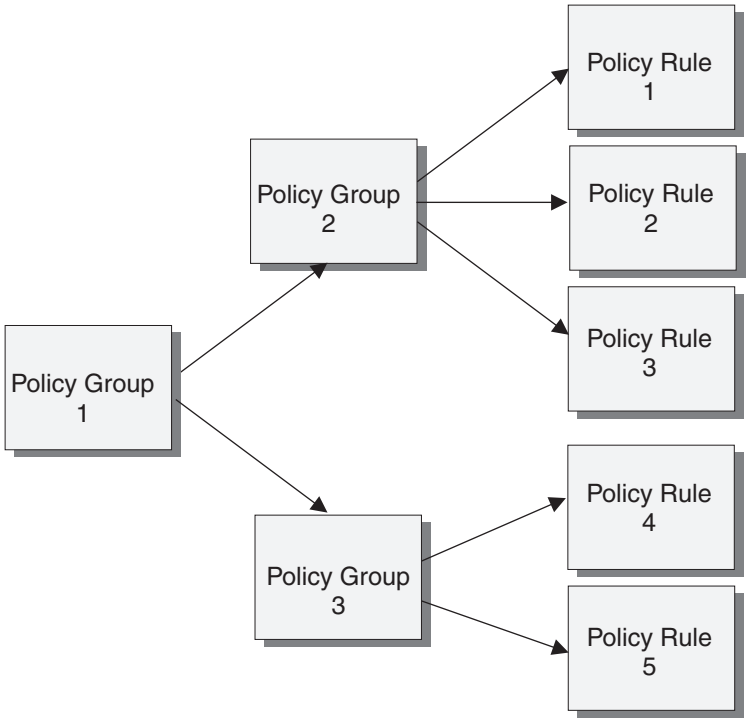


Figure 66. Policy groups

What kind of policy do you want?

The Policy Agent supports the following types of policies:

- Quality of Service (QoS) policies
  - Differentiated Services (DS) policies
  - Integrated Services (RSVP) policies
  - Sysplex Distributor (SD) policies
- Intrusion Detection Services (IDS) policies
  - Scan policies
  - Attack policies
  - Traffic Regulation policies

QoS and IDS policies are defined using different policy schemas. They use a common rule, but have separate conditions and actions. QoS and IDS cannot be mixed in a given policy object. Both QoS and IDS rules may contain time related information that indicates when the policy rule should be considered active or inactive. Active policy rules are installed in the TCP/IP stack, so they can be applied as traffic filters. Inactive policy rules exist only in the Policy Agent.

The Policy Agent supports all of the above policy types, installing them into one or more TCP/IP stacks as configured.

QoS policy

Policy conditions consist of a variety of selection criteria that act as traffic filters. Traffic can be filtered based on source and destination IP addresses, source and destination ports, protocol, inbound and outbound interfaces, application name,

application specific data or application priority. Only packets that match the filter criteria are selected to receive the accompanying action. Policy rules can refer to several policy actions, but only one policy action is executed per policy scope. A given policy action may be referred to by several policy rules.

The type of policy defined is in general controlled by the policy scope value defined for the policy action. SD policies are an exception. SD policies are a subset of DS policies, so use the DS scope.

Although RSVP policies are installed into the TCP/IP stack, they are only used for collecting policy statistics. For policy use and limit enforcement, these policies are requested from the Policy Agent by the RSVP Agent, to apply to RSVP reservation requests from RSVP applications.

IDS policy

Policy conditions primarily determine the portion of IDS function that is being configured. A given IDS policy rule refers to a single IDS policy action. A given IDS policy action may be referred to by several policy rules of the same IDS type. Refer to Chapter 13, “Intrusion Detection Services (IDS)” on page 595 for more details.

Where do you want to define your policies?

Policies can be defined in the Policy Agent Configuration file, in the LDAP server, or both. Policies from both sources are combined into a single list. Note that this requires unique policy object names. For policies defined on the LDAP server, the Distinguished Name (DN) must be unique, but the user-friendly name does not have to be unique (although it is recommended). The Policy Agent appends a unique suffix if required to make LDAP user-friendly names unique within the scope of policies defined on the LDAP server. When policies from a configuration file are combined with LDAP defined policies, the LDAP user-friendly names must be unique with respect to the names defined in the configuration file. Any policy objects with duplicate names at this point are discarded by the Policy Agent. The following table shows some of the advantages and disadvantages of both.

Type	Advantages	Disadvantages
Configuration File	• Policies are easy to define and change. • Definitions are local - no connection needed. • Very little overhead when policies do not change.	• Full power of policy definitions not available (for example can only define simple rules). • Each host must maintain its own policy definitions. • IDS policy is not available.

Type	Advantages	Disadvantages
LDAP Server	• Central policy definitions for many hosts. • Policy definitions can be shared and reused between different platforms and hosts. • Full power of policy definitions available (for example can define complex rules). • Allow robust hierarchical policy definition (Sysplex, LPAR, TCP/IP image). • IDS policy is only available in LDAP.	• Policy definitions are more complex. • LDAP server must be queried at each refresh interval to check for new or changed policies.

LDAP server

Lightweight Directory Access Protocol (LDAP) is a fast-growing technology for accessing common directory information. LDAP has been embraced and implemented in most network-oriented middleware. As an open, vendor-neutral standard, LDAP provides an extendable architecture for centralized storage and management of information that needs to be available for today's distributed systems and services.

After a fast start, it can be assumed that LDAP has become the de facto access method for directory information, much the same as the Domain Name System (DNS) is used for IP address lookup on almost any system on an intranet and on the Internet.

Note: If the z/OS LDAP server is used, a DB2 backend is required.

Overview of the object classes

Policies defined on an LDAP server are comprised of one or more objects, each with a defined object class, and a unique name. Object names are in the form of LDAP Distinguished Names (DNs), which are text strings composed of individual parts known as Relative Distinguished Names (RDNs). The structure of the naming defines a hierarchical tree, in a manner similar to directories in a hierarchical file system. For example, consider the following set of objects:

```
Object 1, DN: o=IBM, c=US
Object 2, DN: cn=group_1, o=IBM, c=US
Object 3, DN: cn=group_5, o=IBM, c=US
Object 4, DN: cn=group_1_sub_A, cn=group_1, o=IBM, c=US
```

This set of objects can be viewed as a tree, with Object 1 as the root. Objects 2 and 3 are branches under the root, with Object 4 a branch under Object 2. The names used are attributes of the objects they define. For example, Object 2, whose name starts with ″cn=group_1″ contains a cn attribute with the value group_1. Refer to *z/OS Security Server LDAP Server Administration and Use* for more information on LDAP naming.

Object class names define the type of each LDAP object. The *top* object class is predefined and is the root of all other object classes.

There are three types of object classes.

Abstract object classes
: Used to define broad concepts, such as policy and to define basic attributes that apply to all subclasses.

Structural object classes
: Basic building blocks, and are the only type of object class that can be instantiated as real objects on an LDAP server.

Auxiliary object classes
: Used to define attributes that are shared among different structural object classes, or are used to extend the basic set of objects.

Attributes from auxiliary classes are *attached* to structural objects by including them in the structural objects, and also by including the auxiliary object class as one of the values of the objectClass attribute in the structural object.

The following object classes are recognized by the Policy Agent. The indentation defines subclasses. For example, ibm-policyGroup is a subclass of ibm-policy, and therefore inherits all of the attributes defined for ibm-policy.

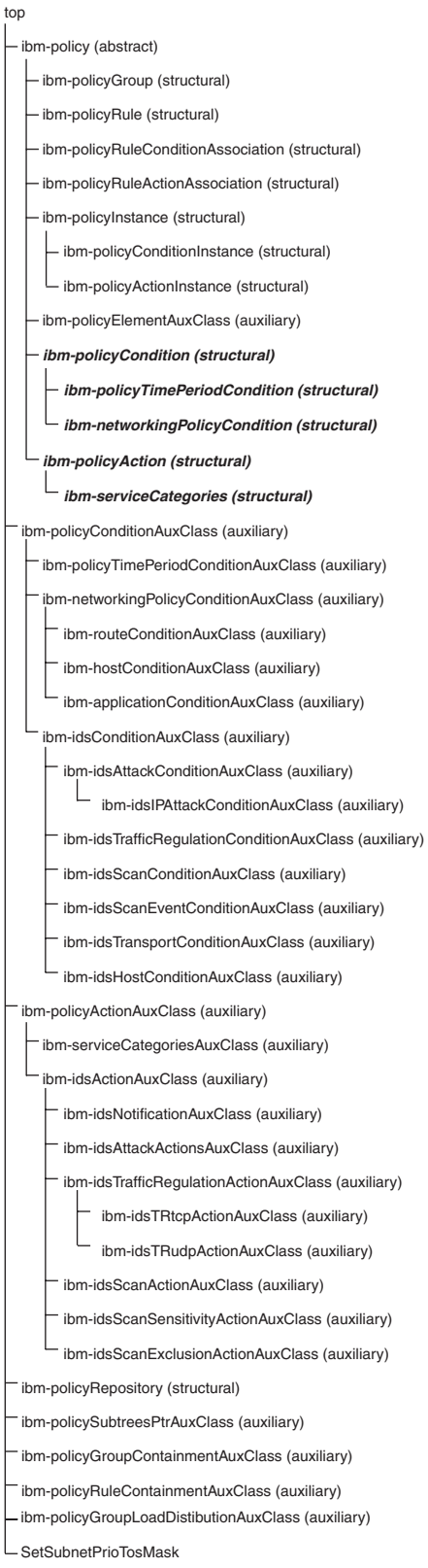


Figure 67. LDAP schema object class hierarchy

Note: The classes identified in italics in the diagram above are for schema version 2 and are mutually exclusive with the schema version 3 classes with similar names.

Object class name	Purpose of object
Top	Used to anchor the LDAP hierarchical tree root.
ibm-policy	Used as the root for all policy objects.
ibm-policyGroup	Defines a policy group object.
ibm-policyRule	Defines a policy rule object.
ibm-policyRuleConditionAssociation	Defines an association between a policy rule object and a policy condition.
ibm-policyRuleActionAssociation	Defines an association between a policy rule object and a policy action.
ibm-PolicyInstance	Defines an instance of a reusable policy object.
ibm-PolicyConditionInstance	Defines an instance of a reusable policy condition object.
ibm-PolicyActionInstance	Defines an instance of a reusable policy action object.
ibm-PolicyElementAuxClass	Defines an auxiliary class that can be used to *tag* non-policy objects as though they were policy objects.
ibm-policyCondition	Defines a policy condition object. (**schema version 2** — supported for migration)
ibm-policyTimePeriodCondition	Defines an auxiliary class to represent time periods during which a policy rule is considered to be active.(**schema version 2** — supported for migration)
ibm-networkingpolicyCondition	Defines a subclass of ibm-PolicyCondition used to define networking policy conditions.(**schema version 2** — supported for migration)
ibm-policyAction	Defines a policy action object.(**schema version 2** — supported for migration)
ibm-serviceCategories	Defines an auxiliary class to represent a set of QoS attributes for a policy action.(**schema version 2** — supported for migration)
ibm-policyConditionAuxClass	Defines an auxiliary class for generic policy conditions.
ibm-policyTimePeriodConditionAuxClass	Defines an auxiliary class to represent time periods during which a policy rule is considered to be active.
ibm-networkingPolicyConditionAuxClass	Defines an auxiliary class used to define networking policy conditions.
ibm-routeConditionAuxClass	Defines an auxiliary class to represent QoS routing conditions for a policy rule.
ibm-hostConditionAuxClass	Defines an auxiliary class to represent QoS host (end-point) conditions for a policy rule.

Object class name	Purpose of object
ibm-applicationConditionAuxClass	Defines an auxiliary class to represent QoS application and transport conditions for a policy rule.
ibm-idsConditionAuxClass	Defines an auxiliary class to represent generic IDS conditions.
ibm-idsAttackConditionAuxClass	Defines an auxiliary class to represent IDS attack conditions.
ibm-idsIPAttackConditionAuxClass	Defines an auxiliary class to represent IDS IP attack conditions.
ibm-idsTrafficRegulationConditionAuxClass	Defines an auxiliary class to represent IDS Traffic Regulation conditions.
ibm-idsScanConditionAuxClass	Defines an auxiliary class to represent IDS global scan conditions.
ibm-idsScanEventConditionAuxClass	Defines an auxiliary class to represent IDS scan event conditions.
ibm-idsTransportConditionAuxClass	Defines an auxiliary class to represent IDS transport conditions.
ibm-idsHostConditionAuxClass	Defines an auxiliary class to represent IDS host conditions.
ibm-policyActionAuxClass	Defines an auxiliary class for generic policy actions.
ibm-serviceCategoriesAuxClass	Defines an auxiliary class to represent a set of QoS attributes for a policy action.
ibm-idsActionAuxClass	Defines an auxiliary class to represent generic IDS actions.
ibm-idsNotificationAuxClass	Defines an auxiliary class to represent notification options for IDS actions.
ibm-idsAttackActionsAuxClass	Defines an auxiliary class to represent attack attributes for IDS actions.
ibm-idsTrafficRegulationActionAuxClass	Defines an auxiliary class to represent generic Traffic Regulation attributes for IDS actions.
ibm-idsTRtcpActionAuxClass	Defines an auxiliary class to represent Traffic Regulation TCP attributes for IDS actions.
ibm-idsTRudpActionAuxClass	Defines an auxiliary class to represent Traffic Regulation UDP attributes for IDS actions.
ibm-idsScanActionAuxClass	Defines an auxiliary class to represent global scan attributes for IDS actions.
ibm-idsScanSensitivityActionAuxClass	Defines an auxiliary class to represent scan sensitivity attributes for IDS actions.
ibm-idsScanExclusionActionAuxClass	Defines an auxiliary class to define scan exclusion lists for IDS actions.
ibm-policyRepository	Defines a repository for generic reusable policy objects.

Object class name	Purpose of object
ibm-policySubtreesPtrAuxClass	Defines an auxiliary class to represent pointers to subtrees in the LDAP directory tree to be retrieved by the Policy Agent. This allows entire subtrees to be retrieved at once, improving retrieval performance in some situations.
ibm-policyGroupContainmentAuxClass	Defines an auxiliary class for binding a policy group object to another policy group.
ibm-policyRuleContainmentAuxClass	Defines an auxiliary class for binding a policy rule object to another policy group.
ibm-policyGroupLoadDistributionAuxClass	Defines an auxiliary class to represent load distribution attributes for policy groups. The load distribution attributes are applied to all policy rules that are pointed to by groups to which this auxiliary class has been attached.
SetSubnetPrioTosMask	Defines a mapping of outbound IP packet Type of Service (ToS) byte values to QDIO device priorities and Virtual LAN (VLAN) user priorities.

Policy objects are used to accomplish the following objectives:

- Group related objects together. Policy groups can contain related policy rules, and can also contain other policy groups. This allows objects to be grouped in various administrative ways. If the resulting objects will be retrieved by any Policy Agent prior to z/OS CS V1R2, then the object should not include the values ibm-policyGroupContainmentAuxClass, ibm-policyRuleContainmentAuxClass or ibm-policyGroupLoadDistributionAuxClass for the objectClass attribute.
- Specify conditions for a policy rule. The conditions are used to filter traffic packets, and can specify attributes such as source and destination port, application name, protocol, and so on. Policy rules can be either simple or complex, depending on the nature of the specified conditions. When a single condition is specified, the rule is a simple rule. This single condition can be composed of any of the condition attributes, but only one instance of each. For example, only a single destination port range can be specified in a simple rule. Complex rules specify more than one condition. The specified conditions are organized into one or more levels, and each level is composed of one or more conditions. Each condition can be composed of one instance of any of the condition attributes. The conditions can thus be thought of as a two-dimensional array. Any individual condition can be negated. Two types of processing are applied to the conditions, depending on the specified condition list type:
  - Disjunctive Normal Form (DNF). DNF conditions are logically ANDed at each level, and ORed between levels.
  - Conjunctive Normal Form (CNF). CNF conditions are logically ORed at each level, and ANDed between levels.

  For example, suppose five conditions are specified, two at one level and three at another:

```
Level 1: C1, NOT C2
Level 2: C3, C4, C5
```

If DNF is specified, the conditions are evaluated as:

```
(C1 AND NOT C2) OR (C3 AND C4 AND C5)
```

CNF evaluation of the same conditions is:

```
(C1 OR NOT C2) AND (C3 OR C4 OR C5)
```

This allows a wide variety of conditional logic to be defined for policy rules.

- Specify time periods during which policy rules are active. Active policy rules are those that are installed in a TCP/IP stack by the Policy Agent. A wide variety of attributes are available to specify time periods, and up to 25 time periods can be specified for any policy rule. The policy rule is active if any of the specified time intervals include the current time.
- Specify actions to take on behalf of traffic that maps to active policy rules, based on the evaluation of its conditions. QoS Actions contain a scope attribute that indicates the type of policy being defined, namely Differentiated Services, RSVP, or both Differentiated Services and RSVP. Up to four actions can be specified for each rule, but only one action per scope can be active at a time. IDS actions contain an action type that indicates the type of policy being defined, namely Attack, TR, Scan Global, or Scan Event. Only one IDS action can be specified for each rule. QoS and IDS actions (or conditions) can't be mixed within a single policy rule.

Considerations for defining LDAP objects

LDAP objects can refer to other objects, using the DN of the referenced object. For example, a policy rule can be separated from its conditions and time periods, with those objects being referenced by the rule object.

Each LDAP object is composed of a number of attributes. Some of the attributes are generic LDAP attributes that apply to all LDAP objects. Other attributes are used only for Version 1 policy definitions. All other Version 2 and later policy attributes must begin with a unique prefix:

```
ibm-
```

When defining complex policy rules (those with more than one condition or action), two mutually exclusive methods can be used to associate the conditions or actions with the rule:

- The ibm-policyConditionListDN and ibm-policyActionListDN attributes can be omitted from the rule. In this case, the condition and action association objects MUST be created as subordinate objects to the policy rule, in other words, *under* the rule in the directory subtree. This is known as Directory Information Tree (DIT)-containment.
- The ibm-policyConditionListDN and ibm-policyActionListDN attributes can be specified in the rule. In this case, the condition and action association objects SHOULD be created as subordinate objects to the policy rule, in other words, *under* the rule in the directory subtree. However, this is not a requirement, only a recommendation. The objects can actually exist anywhere in the DIT.

Policy Agent retrieval of LDAP objects

The design of the LDAP object tree should be carefully thought out. The Policy Agent uses a variety of mechanisms to search for and retrieve objects from an LDAP server:

- An initial search is done for a subtree of objects based on the SearchBaseDN parameter on the ReadFromDirectory statement.

- If any objects retrieved by this initial search contain subtree pointer references (using the ibm-policySubtreesAuxContainedSet attribute) then a search is done for all such subtrees. This is a recursive search: additional objects retrieved might also contain subtree pointer references.
- The above searches use a filter to only retrieve certain object classes. For LDAP protocol version 3, the default is to only scan for the ibm-policy object class. This is an abstract object class from which all other policy object classes are derived. Most LDAPv3 servers implement abstract and auxiliary classes such that this search will properly retrieve policy, and only policy, object classes. However, some LDAPv3 servers do not honor abstract/auxiliary object classes as a search filter. For these servers, specify LDAP_AbstractPolicy NO on the ReadFromDirectory statement. This causes the searches to use a filter that retrieves ALL object classes. This same filter to retrieve all object classes is used for LDAPv2 servers.
- All of the above searches may be scoped, or filtered, using keywords specified on the ReadFromDirectory statement parameters SearchPolicyKeyword, SearchPolicyGroupKeyword, or SearchPolicyRuleKeyword. The LDAP server only returns objects with any matching keywords.
- Some objects retrieved using the above searches may contain DN pointer references to additional objects. These objects are individually retrieved. If the object to be retrieved is a policy rule, then a subtree search is performed, using the keywords specified on the ReadFromDirectory statement. All other objects are retrieved as single objects, using the DN pointers (no keywords are used on the search).
- All policy rule objects retrieved using the above searches are further filtered using the PolicyRole parameter on the ReadFromDirectory statement. Any rules that do not match policy roles specified on the ReadFromDirectory statement are discarded.

Therefore, it is possible to design an LDAP tree such that a minimal set of objects is initially retrieved, followed by many additional individual LDAP retrievals. If the total set of objects is large, there is a performance impact to retrieving objects in this manner. If possible, try to design the tree and the ReadFromDirectory parameters to retrieve the largest set of objects initially, to achieve the best performance, or to use subtree pointer references to retrieve larger sets of objects in one operation.

Installing the schema definition on the LDAP server

The files that define the schema supported by the Policy Agent are shipped as a set of sample files. You need to modify the configuration of the LDAP server to include these schema definition files. How this is accomplished depends on the protocol supported by the server.

For LDAP protocol version 2, include the pagent_oc.conf and pagent_at.conf files (located in the /usr/lpp/tcpip/samples directory) in the server's configuration file. For example:

```
include  /usr/lpp/tcpip/samples/pagent_at.conf
include  /usr/lpp/tcpip/samples/pagent_oc.conf
```

For LDAP protocol version 3, the schema definition is shipped in *ldif* format and installed on the LDAP server as a modification to the generic schema entry, known as a subschema. The existing schema entry must be modified to include the supported schema as a subschema, by using the ldapmodify command. There are

several schema definition files that must be installed in the proper order, as specified in the list below. All of these files are located in the /usr/lpp/tcpip/samples directory):

- pagent_schema.ldif
- pagent_v3schema.ldif
- pagent_schema_updates.ldif
- pagent_idsschema.ldif

Note: This process is supported for the z/OS LDAP server.
Which files need to be installed depends on the current schema definition being used by the LDAP server. For example, the server may already have the pagent_schema.ldif file installed from a previous release. In this case you only need to install the subsequent files in the list above. To install the complete set of schema definitions, use a set of commands like the following example:

```
ldapmodify -h <server address> -p <server port> -D <administrator userid>
-w <password> -f /usr/lpp/tcpip/samples/pagent_schema.ldif
ldapmodify -h <server address> -p <server port> -D <administrator userid>
-w <password> -f /usr/lpp/tcpip/samples/pagent_v3schema.ldif

ldapmodify -h <server address> -p <server port> -D <administrator userid>
-w <password> -f /usr/lpp/tcpip/samples/pagent_schema_updates.ldif

ldapmodify -h <server address> -p <server port> -D <administrator userid>
-w <password> -f /usr/lpp/tcpip/samples/pagent_idsschema.ldif
```

Refer to the TDBM backend information in *z/OS Security Server LDAP Server Administration and Use* for more details.

Using the sample LDAP objects

There are 5 sample files that provide examples of policy definitions in LDAP:

- pagent.ldif
- pagent_starter_IDS.ldif
- pagent_starter_QOS.ldif
- pagent_advanced_IDS.ldif
- pagent_advanced_QOS.ldif

For brief descriptions of these files, see “Policy sample files” on page 541. You can either use some or all of these predefined policies in the starter and advanced sets, or modify them as needed.

The recommended way to create customized policies is to copy the sample policies you want to change to the custom portion of the pagent.ldif file (under the appropriate cn=custom root, QoS or IDS), modify them as needed, and then point to the custom set as the search base on the ReadFromDirectory statement.

For example, the pagent.ldif file has the following hierarchical structure [this shows the relevant parts of the Distingushed Name (DN) for each object]:

```
o=IBM, c=US (root object)
    cn=repository (root of all reusable policy conditions and actions)
    ou=policy (root of all policy groups and rules)
       cn=groups (root of sample groups)
       cn=starter (root of simple starter set of policies)
           cn=IDS (IDS starter set - actually defined in file pagent_starter_IDS.ldif)
           cn=QoS (QOS starter set - actually defined in file pagent_starter_QOS.ldif)
```

```
cn=advanced (root of advanced set of policies)
    cn=IDS (IDS advanced set - actually defined in file pagent_advanced_IDS.ldif)
    cn=QoS (QOS advanced set - actually defined in file pagent_advanced_QOS.ldif)
cn=custom (root of customer-defined set of policies)
    cn=IDS (root of customer-defined IDS policies (empty))
    cn=QoS (root of customer-defined QOS policies (empty))
```

To obtain only the customized policies, specify the top custom policy group object as the search base on the ReadFromDirectory statement as follows:

```
ReadFromDirectory {
...
SearchPolicyBaseDN   dn:cn=custom, ou=policy, o=IBM, c=US
...
}
```

Note: If your LDAP server has a root structure other than ″o=IBM, c=US″, be sure to change the root structure in all the files you want to use by replacing every instance of ″o=IBM, c=US″ with the appropriate root used on your LDAP server.

Policy Agent common functions

Configuring the Policy Agent

Step 1: General configuration

The Policy Agent is responsible for reading policies from a configuration file and/or an LDAP server. Before defining policies, some basic operational characteristics of the Policy Agent need to be configured. Follow these steps to configure these items.

1. Define the TcpImage statements in the main Policy Agent configuration file.

   The Policy Agent can be configured to install policies on one or more TCP/IP stacks, or images. Each TCP/IP stack is configured using a TcpImage statement in the main configuration file. A secondary configuration file can be defined for any given stack, a set of stacks can share configuration information in the main configuration file, or a combination of these techniques can be used.

   To install different sets of policies to different stacks, configure each image with a different secondary configuration file. In this case, each image can be configured with a different policy refresh interval if desired. The refresh interval used for the main configuration file will be the smallest of the values specified for the different stacks.

   Note: When the main configuration file is an MVS data set, it is reread at each refresh interval (which is the smallest of the individual stack refresh intervals), regardless of whether it has actually been changed or not. Because Policy Agent restarts all stack-related processing when the main configuration file is reread, this effectively makes the refresh interval for all stacks the same as this smallest configured interval.

   To install a common set of policies to a set of stacks, do not specify secondary configuration files for each image. In this case, there is only one configuration file (the main one) and the policy information contained in it is installed to all of the configured stacks. Different refresh intervals can also be configured for each image, but would probably be less useful in this case.

In either case, it is possible that TCP/IP stacks configured to the Policy Agent are not started or even defined. The Policy Agent will fail when trying to connect to those stacks and log appropriate error messages.

The Policy Agent does not end when any (or all) stacks end. When the stacks are restarted, active policies are automatically reinstalled.

When the Policy Agent is shutdown normally (KILL or STOP), then if the TcpImage statement option PURGE was coded all policies will be purged from this stack.

The TcpImage statement specifies a TCP/IP image and its associated configuration file to be installed to that image. The following example installs the policy control file */tmp/TCPCS.policy* to the *TCPCS* TCP/IP image, after flushing the existing policy control data:

```
TcpImage TCPCS /tmp/TCPCS.policy FLUSH
```

2. Define the appropriate logging level.

   The LogLevel statement is used to define the amount of information to be logged by the Policy Agent. The default is to log only event, error, console, and warning messages. This might be appropriate for a stable policy configuration, but more information might be required to understand policy processing or debug problems when first setting up policies or when making significant changes. Specify the LogLevel statement with the appropriate logging level in the main configuration file.

   Note: The maximum logging level (511) can produce a significant amount of output, especially with large LDAP configurations. This is not a concern if an HFS log file is used, because Policy Agent uses a set of log files with a finite size in a round-robin configuration (the number and size of these files is controllable with the PAGENT_LOG_FILE_CONTROL environment variable). But when using the syslog daemon as the log file, the amount of log output produced should be taken into consideration.

3. Define security product authorization for the Policy Agent.

   Because the Policy Agent can affect system operation significantly, security product authority (for example, RACF) is required to start the Policy Agent. Refer to the EZARACF sample in SEZAINST for sample commands needed to create the profile name and permit users to it.

4. If Policy Agent's clients (pasearch) are NOT defined as a superuser, then security product authority in the SERVAUTH CLASS for that client MUST be defined to retrieve policies. These profiles can be defined by TCP/IP stack (TcpImage) and policy type (ptype = QoS or IDS). Wildcarding of profile names are allowed.

   ```
   EZB.PAGENT.<sysname>.<TcpImage>.<ptype>
   ```

   where:

   - <sysname> - System name defined in sysplex
   - <TcpImage> - Tcp name for policy information that is being requested
   - ptype - Policy Type that is being requested
     - QOS - Policy QoS
     - IDS - Policy IDS

   Note: Wildcarding is allowed on segments of the profile name.

Policy Agent will check all client's requests to verify SERVAUTH class is active and the profile exists for the TcpImages and policy types in the request. If a client's request is for ALL TcpImages and policy types defined then Policy Agent will only return information for any information for which permission is granted. For example, if the request is for ALL policy types, and both QoS and IDS policy types are defined, but the user is only granted permission for the QoS policy types, then only QoS policy information will be returned.

If SERVAUTH class is absent (not RACLIST) or profiles are absent for client's request (TcpImage, policy type) then permission is denied and data is NOT returned.

If SERVAUTH class is active and profiles are present for client's request (TcpImage and policy type) and MVS user is defined for all profiles then permission is granted and data is returned.

If SERVAUTH class is active and profiles are present for client's request (TcpImage and policy type) and MVS user (that is, Policy Agent client) is not defined for all profiles then permission is refused and data is not returned.

Refer to the EZARACF sample in SEZAINST for sample commands needed to create the profile name and permit users to it.

Step 2: Configure QoS policies in Policy Agent configuration files

QoS policies may be defined in any referenced Policy Agent configuration file. See "Defining policies in a Policy Agent configuration file" on page 570 for more information.

Note: For compatibility, QoS Scope TR policies may also be defined in a Policy Agent configuration file. However, they are transformed internally by Policy Agent into IDS TR policies.

Step 3: Configure Policy Agent to use LDAP server via the ReadFromDirectory statement

The ReadFromDirectory statement in the Policy Agent configuration file initializes the Policy Agent as an LDAP client. The policies are downloaded from the LDAP server, along with the policies specified in the Policy Agent configuration file.

When configuring the ReadFromDirectory statement, first specify the name (or IP address) and port of the primary server and the same for the backup server (if one is used).

Note: When using the z/OS LDAP server, the server listens on a separate port for SSL connections. This means that you should specify the correct port depending on whether or not SSL is used.

Next, configure other connection attributes. The Policy Agent (as an LDAP client) must log in to the LDAP server. *The userid and password for logging in must be configured on the ReadFromDirectory statement.*The userid is also known as Distinguished Name for userid, and it is in the form of an LDAP DN. If the userid and password are not specified, the Policy Agent uses anonymous login to connect to the server.

The LDAP server might be running either Version 2 or Version 3 of the LDAP protocol, which must also be configured on the ReadFromDirectory statement. This statement also configures the version of the schema to be retrieved from the server.

Finally, configure attributes to indicate how to search the LDAP server for policies. Policy roles allow one or more roles, or role-combinations, to be assigned to policy rules using the ibm-policyRoles attribute. These roles represent the intended usage of the policy rules. For example, a role of ″East Coast WAN″ might be used to represent policies for the wide area network on the US East coast for an enterprise. Policy role values are not standardized; they are simply values used to assign roles to policies. When an entity that requires policies (such as Policy Agent) requests policies from an LDAP server, it can filter out policy rules that do not match the roles that it plays. Although similar to policy keywords, which also allow search scoping, policy roles are a bit more sophisticated. Specifically, role-combinations are allowed, which take the form of a specification like ″roleA && roleB″, meaning both roleA AND roleB. Since the ibm-policyRoles attribute is multi-valued, a form of CNF/DNF logic can be used for policy roles: the roles in a role-combination are ANDed, and the roles or role-combinations specified on different values of this attribute are ORed.

For the Version 1 schema, a base DN to start the search, and a *selector tag* value are configured. The selector tag is used to match against the SelectorTag attribute in the policy objects. For Version 1, the Policy Agent also automatically includes the stack name when searching for policies; this value is matched against the TcpImageName attribute in the policy objects. For the Version 2 schema, a base DN to start searching is also configured. This DN can specify a single LDAP object, a policy group, or an LDAP subtree containing many objects. For filtering the search, three keywords can be configured:

- SearchPolicyKeyword matches against the ibm-policyKeywords attribute in any policy object.
- SearchPolicyGroupKeyword matches against the ibm-policyGroupKeywords attribute in policy group objects.
- SearchPolicyRuleKeyword matches against the ibm-policyRuleKeywords attribute in policy rule objects.

The following example does the following:

- Connects to the LDAP server at IP address 9.100.1.1, using the default port 389.
- Specifies a userid and password to log in to the server.
- Specifies LDAP protocol version 3.
- Specifies schema version 3.
- Starts searching at the DN ou=policy, o=IBM, c=US object/subtree.
- Only selects policy objects that contain either the ″POLICY″ or ″EASTERN″ keywords.

```
ReadFromDirectory
{
LDAP_Server 9.100.1.1
LDAP_DistinguishedName cn=root, o=IBM, c=US
LDAP_Password 4qr56jb
LDAP_ProtocolVersion 3
LDAP_SchemaVersion 3
SearchPolicyBaseDN ou=policy, o=IBM, c=US
SearchPolicyKeyword POLICY
SearchPolicyKeyword EASTERN
}
```

Step 4: Optionally add SSL to the Policy Agent connection to LDAP

The Secure Sockets Layer (SSL) protocol begins with a handshake. During the handshake, the client authenticates the server, the server optionally authenticates the client, and the client and server agree on how to encrypt and decrypt information.

Server Authentication: When using SSL to secure communications, the SSL authentication mechanism known as server authentication is used. With server authentication, the LDAP server must have a digital certificate which authenticates the server to the Policy Agent client. The server supplies the client with the certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure communication channel is established between the LDAP server and the Policy Agent client.

For server authentication to work, the LDAP server must have a private key and associated server certificate in the server keyring file.

To conduct commercial business on the Internet, you might use a widely known Certificate Authority (CA), such as VeriSign, to get a high assurance certificate. For a relatively small private network within your own enterprise or group, you can issue your own certificates, called self-signed certificates, for your own use.

Client Authentication: When using SSL Client Authentication, the client passes a digital certificate to the server as part of the SSL handshake. To pass authentication, the Certificate Authority (CA) that signed the client certificate must be considered trusted by the server.

Self-signed Server Certificates: Normally, a server certificate should be obtained from a known CA. However, for testing, an installation might use a self-signed server certificate. Because the clients will not know about the issuer of the self-signed server certificate, in most cases it is necessary to add the server's self-signed certificate to the client's signer certificates.

The gskkyman utility is used to create public/private key pairs and certificate requests, receive certificate requests into a key ring, and manage keys in a keyring. The gskkyman utility is documented in *z/OS System Secure Sockets Layer Programming*. The gskkyman utility is shipped with z/OS in System SSL, which is part of the cryptographic services base element of z/OS.

The Policy Agent connection to LDAP can be secured using SSL by tailoring the following parameters on the ReadFromDirectory statement listed below. This allows for protection of policy retrieval from an LDAP server.

- LDAP_SSLKeyringFile
- LDAP_SSLKeyringPassword
- LDAP_SSLName

For more detail about these parameters, refer to *z/OS Communications Server: IP Configuration Reference*. Additional information about the concepts of cryptography and SSL can be found at the following Web sites:

- http://home.netscape.com/eng/ssl3/
- http://www.verisign.com/repository/crptintr.html

Starting and stopping the Policy Agent

You can start the Policy Agent from the z/OS shell or as a started task. If you use the shell, the Policy Agent should be started in a background shell session, by specifying a trailing & on the command line.

Although the /etc/pagent.conf is the default configuration file, a specific search order is used when starting the Policy Agent. The following order is used:

1. File or data set specified with the -c startup option
2. File or data set specified with the PAGENT_CONFIG_FILE environment variable
3. /etc/pagent.conf
4. hlq.PAGENT.CONF

At initialization, the Policy Agent creates an HFS file called /tmp/*tcpname*.Pagent.tmp. This occurs for every TCP/IP stack defined on a TcpImage statement.

In this HFS file, tcpname is the name of a TCP/IP stack from a TcpImage statement. During TCP/IP stack initialization, the TCP/IP stack will attempt to modify a file by this name to notify the Policy Agent that the stack has been reactivated. This causes the Policy Agent to automatically attempt to reinstall the existing policies to this stack.

To ensure that only one Policy Agent is started, the Policy Agent uses the following enqueue:

- Enqueue name is TCP_TCPI
- Resource name is TCPIP.PAGENT

When starting from the shell, note that the Policy Agent executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your PATH statement contains either /usr/sbin or /usr/lpp/tcpip/sbin.

For example, the following command starts Policy Agent with these characteristics:

```
pagent -c /u/user10/pldap.conf -l SYSLOGD &
```

- Policy Agent uses the configuration file /u/user10/pldap.conf
- Policy Agent logs output to the syslog daemon (SYSLOGD). Note that ″SYSLOGD″ must be specified in uppercase to obtain this behavior

Use the S PAGENT command on an MVS console or SDSF to start the Policy Agent as a started task. A sample procedure is shipped in member EZAPAGSP in SEZAINST.

You can stop the Policy Agent by:

- Using the operator command STOP
- Using the kill command in the z/OS shell
- Using the operator command CANCEL. Use the CANCEL command only as a last resort if the STOP or kill commands do not completely stop the Policy Agent.

Note: When the Policy Agent is shutdown normally (KILL or STOP), then if the TcpImage statement option PURGE was coded all policies will be purged from this stack.

The following kill command with the TERM signal will enable Policy Agent to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where pid is the Policy Agent process ID.

The Policy Agent process ID can be obtained using the following z/OS UNIX command:

```
ps -A
```

It can also be obtained from the /tmp/pagent.pid file. The /tmp/pagent.pid file is a temporary file created by the Policy Agent. It contains the process ID of the current (or last) invocation of the Policy Agent.

Refreshing policies

The MODIFY command may be used to interactively cause the Policy Agent to reread the configuration information and, if requested, download objects from the LDAP server. In addition to this, the Policy Agent will also accept SIGHUP signals to perform the refresh function. Refer to the *z/OS Communications Server: IP System Administrator's Commands* for more detailed information on the MODIFY command.

Verification

To verify that policies are correctly defined and functioning properly, consider the following points:

- Are the policies defined correctly to the LDAP server?
- Are the policies defined correctly to the Policy Agent?

The following sections provide more details about these considerations.

Are the policies defined correctly to the LDAP server?

Refer to the documentation appropriate for the LDAP server which you are using. LDAP servers usually allow you to install multiple files (LDIF), each containing different objects in the LDAP hierarchy. Structural objects higher in the directory tree must be installed before objects that are contained below them. Check for any error messages as each LDIF is installed. Some LDAP servers interpret two consecutive blank lines as end of file. Ensure that all of the objects in the LDIF have been installed by the LDAP server.

Are the policies defined correctly to the Policy Agent?

When starting the Policy Agent, first check for any error messages issued to the console. Message EZZ8434I indicates something is wrong with the Policy Agent environment. Message EZZ8438I indicates a syntax or semantic error in the policy definitions. Messages EZZ8439I and EZZ8440I indicate a problem with the LDAP server configuration or the server itself. Refer to *z/OS Communications Server: IP Diagnosis* for more information on these types of problems. Use the UNIX pasearch command to display policy definitions. The output from this command indicates whether or not policy rules are active, and shows the parsed results of the policy definition attributes. One thing to note is that the Policy Agent is designed to ignore unknown attributes, so misspelled attributes will result in default values being used. The pasearch output can be used to verify that policies are correctly defined.

Chapter 12. Quality of Service (QoS)

Applications and users of TCP/IP networks have different requirements for the service they receive from those networks. A network that treats all traffic as best effort does not meet the needs of such users. *Service differentiation* is a mechanism to provide different service levels to different traffic types based on their requirements and importance in an enterprise network. For example, it might be critical to provide Enterprise Resource Planning (ERP) traffic better service during peak hours than that of FTP or web traffic. The overall service provided to applications or users, in terms of elements such as throughput and delay, is termed *Quality of Service* (QoS). Network service providers that need to provide different QoS levels express their business goals in *Service Level Agreements* (SLAs). There are two types of service in TCP/IP networks that relate to QoS. The first is *Differentiated Services*, which provides QoS to broad classes of traffic or users, for example all outbound web traffic accessed by a particular subnet. The second is *Integrated Services*, which provides end-to-end QoS to an application, by reserving resources along a data path. For z/OS CS, Integrated Services is largely provided by the RSVP Agent, which implements the Resource ReserVation Protocol.

Workload distribution also relates to QoS, in terms of the throughput and delay characteristics of a given server in a sysplex. The ability to dynamically monitor server performance and affect sysplex workload distribution is an important part of the overall QoS of a sysplex. Also important is the ability to limit the set of target systems considered for sysplex routing based on network selection criteria, such as source subnet.

Differentiated Services (DS) policies

Policies for Differentiated Services are used to select and control DS traffic for selected IP servers, such as FTP server traffic. The policy administrator selects the IP traffic to be controlled by defining policy rules. These policy rules include several attributes that can be specified to identify the traffic to be managed. These attributes fall into 2 categories, general attributes and application specified attributes. General attributes can be used to identify the IP traffic of most IP applications using a variety of information, such as:

- The source and/or destination IP addresses or subnets
- The source and destination ports used by the application
- The IP protocol the application is using (TCP or UDP)
- The network interface selected for the outgoing traffic
- The jobname of the application

Application specified attributes allow policy administrators to identify outgoing application IP traffic based on information that is provided and defined by an application. For example, the IBM HTTP Server provides the TCP/IP stack with the URI (Universal Resource Identifier) associated with any outgoing data being sent to a client. This allows the policy administrator to define rules that identify traffic related to specific URIs and policy actions with unique DS controls for this traffic. For example, an installation can define a policy that specifies preferential treatment of outgoing traffic related to the servicing of any URIs beginning with */product/placeorder*. For more information on defining policy rules for the IBM HTTP Server based on URIs refer to *z/OS HTTP Server Planning, Installing, and Using* and the *z/OS Communications Server: IP Configuration Reference*.

Any IP application using the TCP protocol can provide application specified attributes using extensions to the sendmsg() socket API. For more information, refer to the appendix in the *z/OS Communications Server: IP Programmer's Reference*. Application provided attributes can be specified in 2 forms:

- Application defined data: This allows applications to provide free-form text data that can be used to classify the application's outgoing traffic in terms that should be familiar to the application's administrator (for example, URIs are provided by the IBM HTTP server).
- Application specified priority: This allows applications to associate an application priority on the outgoing IP traffic. This application priority in itself does not automatically cause the application's traffic to get preferential treatment. In order to make use of these application specified priorities the policy administrator needs to define policy rules that map these priorities to policy actions that will govern the outgoing traffic of each priority level.

Applications can pass both application defined data and application specified priorities to the TCP/IP stack. When both are specified, the administrator is free to use either or both criteria in their service policy rules. However, it is strongly recommended that any policy rules defined using the application specified attributes should also include at least one general attribute that uniquely identifies the application instance. For example, when defining rules for the HTTP server using URIs, you can help further identify the application by specifying the source port for the server or the HTTP Server's jobname. This will help insure that unauthorized applications cannot exploit policy actions intended for the HTTP Server.

Several aspects of connection and throughput control can be specified with DS policies, including the following:

- TCP connection limits
- Maximum and minimum TCP connection rates, TCP maximum delay
- Committed access bandwidth (mean rate and peak rate) control/enforcement, also known as token bucket traffic shaping
- Type of Service (ToS) byte (also known as DS field - 6–bit value) setting, and mapping to zSeries Queued Direct I/O (QDIO) device priorities and VLAN user priorities.

The above DS service attributes are enforced by the TCP/IP stack in which the DS policies are installed. For additional information on the enforcement of these attributes, refer to *z/OS Communications Server: IP Configuration Reference*.

Token bucket traffic shaping is defined using the following parameters:

- DiffServInProfileRate is the average or mean rate that is desired to be transmitted over time. For example, 256 kilobits per second.
- DiffServInProfileTokenBucket is the *burst* size. This is how much data is allowed to be sent from the application to TCP/IP and still be allowed to be transmitted at the mean rate above. It is suggested, if the application is not policing itself, that this burst size be at least one second's worth of data. Otherwise, if the application is sending large amounts of data at one time to TCP/IP, TCP/IP will slow that application down via congestion windows, and the mean rate may not be achieved.
- DiffServInProfilePeakRate is the highest rate that is allowed to be transmitted for a shorter interval of time. For example, though a customer may only want on average 256 Kb of data per second, they may allow a peak of 512 Kb of data for 1/4 second. The peak rate is used to control the spacing of outbound packets on the transmission line. By having a smaller peak rate, there will be longer spacing

between packets, and thus less burstiness of traffic and increased efficiency. Higher peak rates result in shorter spacing and increased burstiness, which can result in lower link utilization. However, some applications may require it, such as real time or video data.

- DiffServInProfileMaxPacketSize is the amount of data that will be policed at the peak cell rate. For example, if the peak rate is 512 Kb per second, and the maximum packet size is 120 Kb, TCP/IP will only allow about 10 packets of size 1492 bytes to be transmitted every .23 seconds. Again, if an application is sending large amounts of data at one time to TCP/IP, TCP/IP will enforce the peak rate, and anytime more than 10 packets are sent within .25 seconds, TCP/IP will begin to slow this TCP connection. The peak rate can be achieved over a longer period of time if the maximum packet size is entered in larger multiples of packets. However, this will cause greater burstiness as described above. For example, if the maximum packet size is entered as 240 Kb, TCP/IP will allow 20 packets in a .23 second range before enforcing slowdown.

  Note that the peak rate cannot be enforced without mean rate policing. However, you can enforce mean rate without peak rate. Also, setting of these parameters depends on the type of applications and the network that carries it.

Integrated Services (RSVP) policies

RSVP policies are used to set limits on certain parameters requested by RSVP applications. These applications interact with the RSVP Agent to establish resource reservations along a network path, using the RSVP API (RAPI). The reservation requests are in the form of an entity known as a Traffic Specification, or Tspec, which consists of the following values:

- Token bucket mean rate (r)
- Token bucket depth (b)
- Peak rate (p)
- Minimum policed unit (m)
- Maximum packet size (M)

RSVP policies can be used to limit the values requested for (r) and (b), as well as limiting the total number of RSVP reserved flows. The RSVP service attributes are enforced by the RSVP agent which gets RSVP policies from the Policy Agent. For additional information on the enforcement, refer to *z/OS Communications Server: IP Configuration Reference* or RFC 1363.

Sysplex Distributor (SD) policies

Sysplex Distributor policies are used to specify a set of SD target nodes for a given set of traffic. For example, all traffic destined to a given port or application from a specified subnet can be assigned one group of SD target nodes, while traffic for the same port or application from another subnet can be assigned to a different group of target stacks. These policies be can used in conjunction with other Sysplex Distributor controls to assist in load balancing. For more information, see “Policy interactions” on page 262.

QoS specific Policy Agent functions

In addition to supporting the various types of policies, the Policy Agent performs functions related to the Sysplex Distributor. The Policy Agent can be configured to collect network QoS performance data relevant to SD on behalf of policies defined for a target port or application, and assign a *weight fraction* to such policy traffic.

This weight is then used by SD (in conjunction with weights assigned by the Workload Manager) to assist in load balancing decisions. This function is performed by the Policy Agent on SD target nodes within the sysplex.

The Policy Agent also supports *load distribution by service level*. Performance data is kept for each Policy Action (service level) that a target's DVIPA port or application supports. A *Policy Action weight fraction* is generated. This weight, if available, is then used instead of the default (non-service level) *weight fraction* in conjunction with the Workload Manager weight to assist in load distribution decisions for traffic assigned to this service level. If this *Policy Action weight fraction* is unavailable, then the Sysplex Distributor will continue to use the default (non-service level) *weight fraction*.

Another function supported by the Policy Agent is to map Type of Service (ToS) byte values to outbound interface priority values for outbound traffic. The ToS byte is also referred to as the Differentiated Services (DS) byte as an alternative definition (refer to RFC 2474). Note that outbound interface priority values are only meaningful for QDIO interfaces. A set of mappings can be defined to cover various ToS byte values and map them to an appropriate interface priority. All outbound packets over the associated interfaces with a given ToS byte value will then be assigned the corresponding priority value. ToS byte values can also be mapped to Virtual LAN (VLAN) user priorities for propagation over LANs directly connected through OSA-Express.

Sysplex distributor policy performance monitoring configuration

Before activating the sysplex distributor policy performance monitoring function, refer to "Policy interactions" on page 262 for information on workload balancing and policy interactions with sysplex distributor.

The following example illustrates how to activate the policy performance monitoring function for Sysplex Distributor.

Note: This function is activated on SD target stacks and is used to monitor the performance of outbound traffic being serviced by the target stacks. The goal is to detect TCP traffic that exceeds defined thresholds for dropped packets and/or timeouts, and derive a QoS weight fraction for the target stack. This weight fraction is then used to reduce the WLM weight assigned to the target stacks, so that the SD distributing stack can take QoS performance into account.

The following statements apply to the example in this section:

- The policy performance monitoring sampling interval is 60 seconds.
- Policy Agent assigns a loss ratio weight fraction of 25% when the TCP loss ratio (dropped packets to total packets) starts to exceed 2%.
- This weight fraction is increased to 50% when the loss ratio starts to exceed 4%, continuing in this manner up to the maximum loss ratio weight fraction of 95%.
- In a similar manner, a TCP timeout weight fraction of 50% is assigned when the timeout ratio starts to exceed 5%, increasing up to a maximum timeout weight fraction of 100%.
- The individual weight fractions are added together to form a single QoS weight fraction for the target stack, up to a maximum weight fraction of 100%.
- The QoS weight fraction is used at the SD distributing stack to reduce the WLM weight. For example, if the WLM weight is 40, a weight fraction of 50% results in the weight being reduced to 20.

- The traffic to be monitored must be represented by at least one Differentiated Services policy defined for the target application (in this example a policy is defined for Telnet).
- An additional QoS service level fraction is calculated for a target's DVIPA/Port if there are any active connections to the target using that service level.

The fraction is calculated using the PolicyPerfMonitorForSDR parms and is further modified after taking into account the following:

- The number of active connections to this target DVIPA/Port will be compared with the maximum connections allowed for this Policy action.
  - When the number of active connections reaches 70% of maximum connections, then the QoS Policy action fraction will be set to MAX (50%, current calculated value).
  - When the number of active connections reaches 85% of maximum connections, then the QoS Policy action fraction will be set to MAX (85%, current calculated value).
  - When the number of active connections reaches 95% of maximum connections, then the QoS Policy action fraction will be set to 100%.
- The throughput rate for this timer interval will be calculated and compared to the DiffServ mean rate of this Policy action. If the throughput rate is greater than 85% of the DiffServ mean rate, then the average throughput rate per connection will be calculated. If the throughput rate per connection is less than the DiffServ min rate, then the minimum throughput requirement/connection is not being met; the QoS Policy action fraction will be set to 100%.
- Only one policy rule and policy action are defined here.

As a result, only Telnet QoS performance information is monitored by the Policy Agent for Sysplex Distributor to route incoming Telnet connections to this target node relative to other target nodes which presumably can also accept Telnet requests.

```
PolicyPerfMonitorForSDR  enable
{
   samplinginterval 60
   LossRatioAndWeightFr 20 25
   TimeoutRatioAndWeightFr 50 50
   LossMaxWeightFr 95
   TimeoutMaxWeightFr 100
}
policyAction telnetGold
{
MinRate 500 # Provide minimum rate of 500 Kbps.
OutgoingTOS 10100000 # the TOS value of outgoing telnet packets.
}

policyRule targetelnet
{
ProtocolNumberRange 6
SourcePortRange 23
policyactionreference telnetGold
}
```

Type of Service (ToS) mapping configuration

There are two mappings provided by the SetSubnetPrioTosMask statement:

- ToS to device priority

  Quality of Service (QoS) support in z/OS CS allows the ToS byte, also known as the Differentiated Services (DS) field, to be set for outbound IP packets according

to defined policies managed by the z/OS CS UNIX Policy Agent. When IP packets are sent out over QDIO devices, the ToS/DS value is mapped to a QDIO priority value. Device priority values are 1-4, where 1 is the highest priority.

- ToS to VLAN user priority

  ToS/DS values can be mapped to user priorities for directly attached LANs using OSA-Express in QDIO mode. VLAN user priority values are 0-7, where 7 is the highest priority. This allows assigned user priorities to be propagated through such networks, resulting in no loss of priority information for data being served by z/OS.

Refer to the *z/OS Communications Server: IP Configuration Reference* for more detail on these statements.

The following example shows a mapping of various ToS byte values to associated interface priority values. Note that the mapping can be applied to individual interfaces or all interfaces:

- The first example defines a mapping for a specific interface. Note that the specified interface must be a valid interface specified in the HOME list. The second example shows a different mapping for all other interfaces.
- The subnet mask defines the bits in the ToS byte that are significant. These examples use the leftmost 3 bits.
- The first example shows a set of mappings defining the complete set of ToS byte values and the device and VLAN user priorities to be assigned for each value.
- The second example shows a set of mappings defining the complete set of ToS byte values and the device priority to be assigned for each value.

```
SetSubnetPrioTosMask
{
  SubnetAddr         10.10.1.5
  SubnetTosMask      11100000
  PriorityTosMapping 1 11100000 7
  PriorityTosMapping 1 11000000 7
  PriorityTosMapping 2 10100000 6
  PriorityTosMapping 2 10000000 5
  PriorityTosMapping 2 01100000 5
  PriorityTosMapping 3 01000000 3
  PriorityTosMapping 4 00100000 2
  PriorityTosMapping 4 00000000 0
}
SetSubnetPrioTosMask
{
  SubnetTosMask      11100000
  PriorityTosMapping 1 11100000
  PriorityTosMapping 1 11000000
  PriorityTosMapping 1 10100000
  PriorityTosMapping 1 10000000
  PriorityTosMapping 2 01100000
  PriorityTosMapping 2 01000000
  PriorityTosMapping 3 00100000
  PriorityTosMapping 4 00000000
}
```

Defining policies in a Policy Agent configuration file

Configure the following statements in the configuration file to define policies:

- PolicyAction
- PolicyRule

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these statements.

The following sections contain examples of these tasks.

Note: These examples are for illustrative purposes only. The policies deliberately use a wide variety of attributes, and they do not necessarily represent real-world usage. Some examples show continued and indented statements that were modified to fit on the page and therefore are not an actual representation of proper syntax.

Differentiated Services policy examples

The goal of this Differentiated Services policy is to map a subset of the traffic outbound from an FTP server.

This policy is identified as a Differentiated Services policy by the PolicyScope DataTraffic attribute on the PolicyAction statement, as well as the use of several DS-only attributes.

The following statements apply to the example in this section:

- The policy rule selects traffic originated by ports in the range 20-21 (FTP outbound data connection uses port 20) from the source address 200.50.23.11.
- The policy rule is active on weekdays between 6 a.m. and 10 p.m. local time, between the dates 7/1/2000 and 7/1/2005..
- The policy action specifies that the ToS byte be set to '10000000' for traffic that conforms to this policy.
- The action establishes a *token bucket* traffic conditioner with a mean rate of 256 kilobits per second, a peak rate of 512 kilobits per second, and a burst size of 64 kilobytes. Any traffic that exceeds these specifications will be sent as *best effort*, with an accompanying ToS byte of '00000000'.

```
PolicyRule                              diffServ
    {
       ProtocolNumberRange                  6
       SourceAddressRange                   200.50.23.11
       SourcePortRange                      20-21
       PolicyActionReference                tokenbucket
       PolicyRulePriority                   10
       ConditionTimeRange                   20000701000000:20050630235959
       DayOfMonthMask                       11111111111111111111111111111111
       DayOfWeekMask                        0111110
       TimeOfDayRange                       06:00-22:00
    }
    PolicyAction                            tokenbucket
    {
       PolicyScope                          DataTraffic
       OutgoingTOS                          10000000
       DiffServInProfileRate                256       # 256 Kbps
       DiffServInProfileTokenBucket         512       # 512 Kbits
       DiffServInProfilePeakRate            512       # 512 Kbps
       DiffServInProfileMaxPacketSize       120       # 120 Kbits
       DiffServOutProfileTransmittedTOSByte 00000000
       DiffServExcessTrafficTreatment       BestEffort
    }
```

The goal of this policy is to ensure that outgoing data that match the specified attributes will be assigned a QoS service level defined in action ″interactive1″.

The following statements apply to the example in this section:

- This rule will only match traffic on TCP connections (protocol 6) with a source port of 80 (i.e. HTTP server) and application defined data beginning with the string ″/catalog″.
- Since we are dealing with HTTP traffic, this rule is basically indicating that all outgoing traffic associated with a URI that begins with ″/catalog″ should be managed using the DS characteristics specified in the ″interactive1″ policy action.

```
PolicyRule          web-catalog       # web catalog traffic
{
     protocolNumberRange  6
     SourcePortRange    80
     ApplicationData /catalog
     policyActionReference  interactive1

}

PolicyAction      interactive1
{
    policyScope DataTraffic
    outgoingTOS 10000000
}
```

RSVP policy example

The goal of this RSVP policy is to establish limits on resource reservations requested by RSVP applications using the RSVP API (RAPI) interface. The policy is identified as an RSVP policy by the PolicyScope attribute on the PolicyAction statement, as well as the use of RSVP-only attributes.

The following statements apply to the example in this section:

- The policy rule selects traffic from source ports in the range 8000 to 8001, with a protocol ID of 6 (TCP).
- The DataTraffic policy action specifies that the ToS byte be set to 01100000 for differentiated services traffic that conforms to this policy. Essentially, any traffic sent by the target application without an RSVP reservation in place will use this policy action. Once an RSVP reservation is in place, the RSVP action gets used.
- The RSVP policy action specifies that the ToS byte be set to 01100000 while an RSVP reservation is in place. It also limits the type of RSVP service requested by RSVP applications to Controlled Load. Applications requesting Guaranteed service are downgraded to using Controlled Load service. In addition, the action limits the mean rate and token bucket size to 50000 bytes per second and 6000 bytes, respectively. These values are requested by RSVP applications in the traffic specification, or Tspec.
- The action also limits the number of active RSVP flows that map to this policy to 10.

```
PolicyRule                              intserv
     {
        SourcePortRange                      8000 8001
        ProtocolNumberRange                  6
        PolicyActionReference                intserv1
        PolicyActionReference                intserv2
     }
     PolicyAction                            intserv1
     {
        PolicyScope                          DataTraffic
        OutgoingTOS                          01100000
     }
     PolicyAction                            intserv2
     {
        PolicyScope                          RSVP
        OutgoingTOS                          01100000
```

```
        FlowServiceType                     ControlledLoad
        MaxRatePerFlow                      400    # 50000 bytes/second
        MaxTokenBucketPerFlow               48     # 6000  bytes
        MaxFlows                            10
    }
```

Sysplex Distributor policy example

The goal of this Sysplex Distributor policy is to limit the number of SD target stacks for inbound Telnet traffic. The policies are identified as SD policies by the ForLoadDistribution TRUE attribute on the PolicyRule statement. The corresponding policy on the target is also shown.

The following statements apply to the example in this section:

- Separate policies are defined on the Sysplex Distributor distributing and target stacks.
- The policy rules select incoming Telnet connection requests.
- The selected target stack will be based on WLM information and QoS information if activated at the target stacks.
- The rule (disttelnet) is coded on the distributing stack to select inbound traffic destined to the Telnet server.
- The rule (targtelnet) is coded on the target stack to select outbound data from the Telnet server.
- If none of the specified target stacks is available to service incoming requests (either the node is down or the Telnet server is not active), then Sysplex Distributor will distribute the requests to any available target stack.

  Note: If the OutboundInterface 0.0.0.0 statement were not present, and neither of the defined target stacks were available, Sysplex Distributor would reject the request.

```
policyAction        telnetGold
    {
        MinRate         500       # Provide minimum rate of 500 Kbps.
        OutgoingTOS     10100000  # the TOS value of outgoing telnet packets.
        outboundinterface  129.100.11.1
        outboundinterface  129.100.21.1
        outboundinterface  129.200.12.1
        outboundinterface  0.0.0.0
    }

    policyRule           disttelnet
    {
        ProtocolNumberRange     6
        DestinationPortRange    23
        PolicyRulePriority      20
        policyactionreference   telnetGold
        ForLoadDistribution     TRUE
    }

    policyRule           targtelnet
    {
        ProtocolNumberRange     6
        SourcePortRange         23
        PolicyRulePriority      20
        policyactionreference   telnetGold
        ForLoadDistribution     FALSE
    }
```

Notes:

1. The ApplicationName attribute is only valid for a target rule and should not be coded on a distributor rule because the application name determined for inbound traffic (which is always the case on a distributor) will always be the stack's TCP jobname.
2. If you are using Telnet with multiple stacks in conjunction with the Sysplex Distributor, see Chapter 8, "Accessing remote hosts using Telnet" on page 305 for more information.

Defining policies using LDAP

Differentiated Services policy example

The goal of this Differentiated Services policy is to map a subset of the traffic outbound from an FTP server.

This policy is identified as a Differentiated Services policy by the ibm-PolicyScope:DataTraffic attribute in the ibm-PolicyActionInstance object.

The following statements apply to the example in this section:

- The policy rule selects traffic originated by ports in the range 20-21 (FTP outbound data connection uses port 20) from the source address 200.50.23.11 or 200.50.33.14 or 202.9.55.1.
- The policy rule is active on weekdays between 6 a.m. and 10 p.m. local time, between the dates 7/1/2000 and 7/1/2005.
- The policy action specifies that the ToS byte be set to '10000000' for traffic that conforms to this policy.
- The action establishes a *token bucket* traffic conditioner with a mean rate of 256 kilobits per second, a peak rate of 512 kilobits per second, and a burst size of 64 kilobytes. Any traffic that exceeds these specifications will be sent as *best effort*, with an accompanying ToS byte of '00000000'.

```
dn:cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:diffserv-rule
ibm-policyRuleName:diffserv-rule
ibm-policyRuleEnabled:1
ibm-policyRuleConditionListType:2
ibm-policyRuleConditionListDN:cn=condassoc1, cn=diffserv-rule, cn=QoS, cn=advanced,
    ou=policy, o=IBM, c=US
ibm-policyRuleConditionListDN:cn=condassoc2, cn=diffserv-rule, cn=QoS, cn=advanced,
    ou=policy, o=IBM, c=US
ibm-policyRuleConditionListDN:cn=condassoc3a, cn=diffserv-rule, cn=QoS, cn=advanced,
    ou=policy, o=IBM, c=US
ibm-policyRuleConditionListDN:cn=condassoc3b, cn=diffserv-rule, cn=QoS, cn=advanced,
    ou=policy, o=IBM, c=US
ibm-policyRuleConditionListDN:cn=condassoc3c, cn=diffserv-rule, cn=QoS, cn=advanced,
    ou=policy, o=IBM, c=US
ibm-policyRuleActionListDN:cn=actassoc1, cn=diffserv-rule, cn=QoS, cn=advanced,
    ou=policy, o=IBM, c=US
ibm-policyRuleValidityPeriodList:cn=period1, cn=time, cn=repository, o=IBM, c=US
ibm-policyRulePriority:10
ibm-policyRuleMandatory:TRUE
ibm-policyRuleSequencedActions:1
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:QOS Differentiated Services rule

dn:cn=condassoc1, cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
```

```
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
ibm-policyConditionName:diffserv-condition1
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=IpProtTCP, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents reusable CNF condition at level 1 - TCP

dn:cn=condassoc2, cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc2
ibm-policyConditionName:diffserv-condition2
ibm-policyConditionGroupNumber:2
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=ftpdPorts, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents reusable CNF condition at level 2 - ftpd ports

dn:cn=condassoc3a, cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc3a
ibm-policyConditionName:diffserv-condition3a
ibm-policyConditionGroupNumber:3
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=Host1, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents first reusable CNF condition at level 3 - host1

dn:cn=condassoc3b, cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc3b
ibm-policyConditionName:diffserv-condition3b
ibm-policyConditionGroupNumber:3
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=Host2, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents second reusable CNF condition at level 3 - host2

dn:cn=condassoc3c, cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc3c
ibm-policyConditionName:diffserv-condition3c
ibm-policyConditionGroupNumber:3
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=Host3, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents third reusable CNF condition at level 3 - host3

dn:cn=actassoc1, cn=diffserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:diffserv-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=tokenbucket, cn=QoSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
```

```
ibm-policyKeywords:POLICY
description:Represents reusable action - token bucket

dn: cn=tokenbucket, cn=QoSact, cn=repository, o=IBM, c=US
objectclass:ibm-policyActionInstance
objectclass:ibm-policyActionAuxClass
objectclass:ibm-serviceCategoriesAuxClass
cn:tokenbucket
ibm-policyActionName:tokenbucket-action
ibm-PolicyScope:DataTraffic
ibm-OutgoingTOS:10000000
ibm-DiffServInProfileRate:256
ibm-DiffServInProfilePeakRate:512
ibm-DiffServInProfileTokenBucket:512
ibm-DiffServInProfileMaxPacketSize:120
ibm-DiffServOutProfileTransmittedTOSByte:00000000
ibm-DiffServExcessTrafficTreatment:BestEffort
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS diffserv token bucket action

dn:cn=ftpdPorts, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:ftpdPorts
ibm-policyConditionName:ftpdPorts-condition
ibm-sourcePortRange:20-21
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS ftpd ports condition

dn:cn=IpProtTCP, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:IpProtTCP
ibm-policyConditionName:IpProtTCP-condition
ibm-protocolNumberRange:6
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS protocol TCP condition

dn:cn=Host1, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-hostConditionAuxClass
cn:Host1
ibm-policyConditionName:Host1-condition
ibm-SourceIPAddressRange:3-200.50.23.11
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS Host 1 condition

dn:cn=Host2, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-hostConditionAuxClass
cn:Host2
ibm-policyConditionName:Host2-condition
ibm-SourceIPAddressRange:3-200.50.33.14
ibm-policyKeywords:QOS
```

```
ibm-policyKeywords:POLICY
description:Reusable QoS Host 2 condition

dn:cn=Host3, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-hostConditionAuxClass
cn:Host3
ibm-policyConditionName:Host3-condition
ibm-SourceIPAddressRange:3-202.9.55.1
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS Host 3 condition

dn:cn=period1, cn=time, cn=repository, o=IBM, c=US
objectclass:ibm-policyInstance
objectclass:ibm-policyTimePeriodConditionAuxClass
cn:period1
ibm-policyInstanceName:WeekdayPrime-time
ibm-ptpConditionTime:20000701000000:20050630235959
ibm-ptpConditionMonthOfYearMask:111111111111
ibm-ptpConditionDayOfMonthMask:11111111111111111111111111111111
ibm-ptpConditionDayOfWeekMask:0111110
ibm-ptpConditionTimeOfDayMask:060000:220000
ibm-ptpConditionLocalOrUtcTime:1
ibm-policyKeywords:POLICY
description:Active weekdays 6am - 10pm local time, 7/1/2000 to 7/1/2005
```

The goal of this policy is to ensure that outgoing data that match the specified attributes will be assigned a QoS service level defined in action ″interactive1″.

The following statements apply to the example in this section:

- This rule will only match traffic on TCP connections (protocol 6) with a source port of 80 (i.e. HTTP server) and application defined data beginning with the string ″/catalog″.
- Since we are dealing with HTTP traffic, this rule is basically indicating that all outgoing traffic associated with a URI that begins with ″/catalog″ should be managed using the DS characteristics specified in the ″interactive1″ policy action.

```
dn:cn=web-catalog-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:web-catalog-rule
ibm-policyRuleName:web-catalog-rule
ibm-policyRuleEnabled:1
ibm-policyRuleConditionListType:1
ibm-policyRulePriority:10
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:QOS Web catalog rule

dn:cn=condassoc1, cn=web-catalog-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
ibm-policyConditionName:web-catalog-condition1
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=IpProtTCP, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents first reusable DNF condition - TCP

dn:cn=condassoc2, cn=web-catalog-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
```

```
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc2
ibm-policyConditionName:web-catalog-condition2
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=webPort, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents second reusable DNF condition - web port

dn:cn=condassoc3, cn=web-catalog-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:condassoc3
ibm-policyConditionName:web-catalog-condition3
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-applicationData:/catalog
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Rule-specific condition - web catalog pages

dn:cn=actassoc1, cn=web-catalog-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:web-catalog-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=interactive1, cn=QoSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents reusable action - interactive 1

dn: cn=interactive1, cn=QoSact, cn=repository, o=IBM, c=US
objectclass:ibm-policyActionInstance
objectclass:ibm-policyActionAuxClass
objectclass:ibm-serviceCategoriesAuxClass
cn:interactive1
ibm-policyActionName:interactive1-action
ibm-policyScope:DataTraffic
ibm-outgoingTOS:10000000
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS interactive 1 action (TOS 100)

dn:cn=webPort, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:webPort
ibm-policyConditionName:webPort-condition
ibm-sourcePortRange:80
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS web port condition

dn:cn=IpProtTCP, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:IpProtTCP
```

```
ibm-policyConditionName:IpProtTCP-condition
ibm-protocolNumberRange:6
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS protocol TCP condition
```

RSVP policy example

The goal of this RSVP policy is to establish limits on resource reservations requested by RSVP applications using the RSVP API (RAPI) interface. The policy is identified as an RSVP policy by the ibm-PolicyScope:RSVP attribute in the ibm-PolicyActionInstance object.

The following statements apply to the example in this section:

- The policy rule selects traffic from source ports in the range 8000 to 8001, with a protocol ID of 6 (TCP).
- One policy action specifies that the ToS byte be set to 01100000 for traffic that conforms to this policy.
- The RSVP policy action limits the type of RSVP service requested by RSVP applications to Controlled Load. Applications requesting Guaranteed service are downgraded to using Controlled Load service. In addition, the action limits the mean rate and token bucket size to 50000 bytes per second and 6000 bytes, respectively. These values are requested by RSVP applications in the traffic specification, or Tspec.
- The action also limits the number of active RSVP flows that map to this policy to 10.

```
dn:cn=intserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:intserv-rule
ibm-policyRuleName:intserv-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleConditionListDN:cn=condassoc1, cn=intserv-rule, cn=QoS, cn=advanced,
     ou=policy, o=IBM, c=US
ibm-policyRuleActionListDN:cn=actassoc1, cn=intserv-rule, cn=QoS, cn=advanced,
     ou=policy, o=IBM, c=US
ibm-policyRuleActionListDN:cn=actassoc2, cn=intserv-rule, cn=QoS, cn=advanced,
     ou=policy, o=IBM, c=US
ibm-policyRuleValidityPeriodList:cn=period2, cn=time, cn=repository, o=IBM, c=US
ibm-policyRuleValidityPeriodList:cn=period3, cn=time, cn=repository, o=IBM, c=US
ibm-policyKeywords:Intserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:QOS Integrated Services rule

dn:cn=condassoc1, cn=intserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-hostConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:condassoc1
ibm-policyConditionName:intserv-condition
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-ProtocolNumberRange:6
ibm-SourceIPAddressRange:1
ibm-SourcePortRange:8000-8001
ibm-policyKeywords:Intserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Rule-specific condition - all local IP addresses, application TCP ports
```

```
dn:cn=actassoc1, cn=intserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
objectclass:ibm-policyActionAuxClass
objectclass:ibm-serviceCategoriesAuxClass
cn:actassoc1
ibm-policyActionName:intserv-action1
ibm-policyActionOrder:1
ibm-PolicyScope:DataTraffic
ibm-OutgoingTOS:01100000
ibm-policyKeywords:Intserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Rule-specific action - set TOS

dn:cn=actassoc2, cn=intserv-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
objectclass:ibm-policyActionAuxClass
objectclass:ibm-serviceCategoriesAuxClass
cn:actassoc2
ibm-policyActionName:intserv-action2
ibm-policyActionOrder:2
ibm-PolicyScope:RSVP
ibm-OutgoingTOS:01100000
ibm-FlowServiceType:ControlledLoad
ibm-MaxRatePerFlow:400
ibm-MaxTokenBucketPerFlow:48
ibm-MaxFlows:10
ibm-policyKeywords:Intserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Rule-specific action - RSVP limitations

dn:cn=period2, cn=time, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyTimePeriodConditionAuxClass
cn:period2
ibm-policyConditionName:EndOfMonth-time
ibm-ptpConditionDayOfMonthMask:00000000000000000000000000000001
     00000000000000000000000000000000
ibm-ptpConditionLocalOrUtcTime:2
ibm-policyKeywords:POLICY
description:Active last day of the month (in UTC)

dn:cn=period3, cn=time, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyTimePeriodConditionAuxClass
cn:period3
ibm-policyConditionName:PacificNight-time
ibm-ptpConditionTimeOfDayMask:190000:030000
ibm-ptpConditionTimeZone:-08
ibm-policyKeywords:POLICY
description:Active 7pm - 3am local time, Pacific Time Zone (no daylight savings)
```

Sysplex Distributor routing policy example

The goal of this Sysplex Distributor policy is to limit the number of SD target stacks for inbound Telnet traffic. The policies are identified as SD policies by the ibm-policyGroupForLoadDistribution:TRUE attribute in the ibm-PolicyGroup object.

The following statements apply to the example in this section:

- Separate policies are defined on the Sysplex Distributor distributing and target stacks.
- The policy rules select incoming Telnet connection requests.

- The selected target stack will be based on WLM information and QoS information if activated at the target stacks.
- The rule (disttelnet) is coded on the distributing stack to select inbound traffic destined to the Telnet server.
- The rule (targtelnet) is coded on the target stack to select outbound data from the Telnet server.
- If none of the specified target stacks is available to service incoming requests (either the node is down or the Telnet server is not active), then Sysplex Distributor will distribute the requests to any available target stack.

Note: If the ibm-Interface:1–0.0.0.0 attribute were not present, and none of the defined target stacks were available, Sysplex Distributor would reject the request.

```
dn:cn=sysplex, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyGroup
objectclass:ibm-policyRuleContainmentAuxClass
objectclass:ibm-policyGroupLoadDistributionAuxClass
cn:sysplex
ibm-policyGroupName:QoSadvancedsysplex-Group
ibm-policyRulesAuxContainedSet:cn=disttelnet-rule,cn=QoS,cn=advanced,ou=policy,
     o=IBM,c=US
ibm-policyGroupForLoadDistribution:TRUE
ibm-policyKeywords:Sysplex
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description: QoS advanced examples sysplex group.

dn:cn=disttelnet-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:disttelnet-rule
ibm-policyRuleName:disttelnet-rule
ibm-policyRuleValidityPeriodList:cn=period1, cn=time, cn=repository, o=IBM, c=US
ibm-policyRulePriority:20
ibm-policyKeywords:Sysplex
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:QOS Sysplex Distributor telnet rule

dn:cn=condassoc1, cn=disttelnet-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:condassoc1
ibm-policyConditionName:disttelnet-condition
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-ProtocolNumberRange:6
ibm-DestinationPortRange:23
ibm-policyKeywords:Sysplex
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Rule-specific condition - telnet inbound SD traffic

dn:cn=actassoc1, cn=disttelnet-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:disttelnet-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=telnetGold, cn=QoSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Sysplex
```

```
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents reusable action - telnet Gold Service

dn:cn=targtelnet-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:targtelnet-rule
ibm-policyRuleName:targtelnet-rule
ibm-policyRuleConditionListType:2
ibm-policyRuleValidityPeriodList:cn=period1, cn=time, cn=repository, o=IBM, c=US
ibm-policyRulePriority:20
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:QOS Sysplex Target telnet rule

dn:cn=condassoc1, cn=targtelnet-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
ibm-policyConditionName:targtelnet-condition1
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=IpProtTCP, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents reusable condition at level 1 - TCP

dn:cn=condassoc2, cn=targtelnet-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc2
ibm-policyConditionName:targtelnet-condition2
ibm-policyConditionGroupNumber:2
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=telnetdPort, cn=QoScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents reusable condition at level 2 - telnetd port

dn:cn=actassoc1, cn=targtelnet-rule, cn=QoS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:targtelnet-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=telnetGold, cn=QoSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Represents reusable action - telnet Gold Service

dn: cn=telnetGold, cn=QoSact, cn=repository, o=IBM, c=US
objectclass:ibm-policyActionInstance
objectclass:ibm-policyActionAuxClass
objectclass:ibm-serviceCategoriesAuxClass
cn:telnetGold
ibm-policyActionName:telnetGold-action
ibm-PolicyScope:DataTraffic
ibm-OutgoingTOS:10100000
ibm-MinRate:500
ibm-Interface:1--129.100.11.1
ibm-Interface:1--129.100.21.1
ibm-Interface:1--129.200.12.1
ibm-Interface:1--0.0.0.0
ibm-policyKeywords:Diffserv
ibm-policyKeywords:QOS
```

```
ibm-policyKeywords:POLICY
description:Reusable QoS telnet Gold Service action

dn:cn=telnetdPort, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:telnetdPort
ibm-policyConditionName:telnetdPort-condition
ibm-sourcePortRange:23
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS telnetd port condition

dn:cn=IpProtTCP, cn=QoScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-policyConditionAuxClass
objectclass:ibm-networkingPolicyConditionAuxClass
objectclass:ibm-applicationConditionAuxClass
cn:IpProtTCP
ibm-policyConditionName:IpProtTCP-condition
ibm-protocolNumberRange:6
ibm-policyKeywords:QOS
ibm-policyKeywords:POLICY
description:Reusable QoS protocol TCP condition

dn:cn=period1, cn=time, cn=repository, o=IBM, c=US
objectclass:ibm-policyInstance
objectclass:ibm-policyTimePeriodConditionAuxClass
cn:period1
ibm-policyInstanceName:WeekdayPrime-time
ibm-ptpConditionTime:20000701000000:20050630235959
ibm-ptpConditionMonthOfYearMask:111111111111
ibm-ptpConditionDayOfMonthMask:11111111111111111111111111111111
ibm-ptpConditionDayOfWeekMask:0111110
ibm-ptpConditionTimeOfDayMask:060000:220000
ibm-ptpConditionLocalOrUtcTime:1
ibm-policyKeywords:POLICY
description:Active weekdays 6am - 10pm local time, 7/1/2000 to 7/1/2005
```

RSVP

Resource ReSerVation Protocol (RSVP) is a protocol that provides a mechanism to reserve resources in support of Integrated Services. The z/OS UNIX RSVP agent provides the following services on behalf of applications that want to use Integrated Services:

- An RSVP API (RAPI) that allows applications to explicitly request RSVP services. Using RAPI, applications indicate their intent to send or receive data, describe the characteristics of the data traffic and request that RSVP reserve resources along the data path to provide a given QoS to one or more traffic flows. For more information about RAPI, refer to *z/OS Communications Server: IP Programmer's Reference*.
- Mapping of IP ToS settings to RSVP traffic, using policies defined for RSVP.
- Establishment of resource reservations on ATM interfaces by use of reserved SVC connections.

  Note: Resource reservations cannot be made on interfaces other than ATM for outbound traffic on z/OS. However, RSVP-capable routers in the network can still reserve resources, and the ToS byte can be set for RSVP traffic to provide further means of prioritizing traffic.

- Support for VIPA addresses as well as real IP addresses.
- Communication with other RSVP agents on hosts and routers in the network to communicate application resource reservation requests.

Network administrators can use the z/OS UNIX Policy Agent to define RSVP-specific policies. These policies can be used to limit the parameters of application-requested resource reservations, provide ToS mappings for RSVP traffic, and limit the number of traffic flows that can use RSVP services simultaneously.

RSVP is designed to be implemented on both end systems (hosts) and routers. Different functions are provided by RSVP in these two environments. The z/OS RSVP agent is supported as a host RSVP implementation only. It can communicate with router RSVP implementations, but is not itself supported as such. For more information about RSVP, refer to RFC 2205.

Configuring the RSVP agent

To configure the RSVP agent, update the configuration file to specify RSVP agent operational parameters using the LogLevel, TcpImage, Interface and RSVP statements. Refer to *z/OS Communications Server: IP Configuration Reference* for detailed information about the statements.

To start the RSVP agent, you must first authorize the RSVP Agent using the security product. See SEZAINST(EZARACF) for SAF considerations for started tasks.

The following is an example of an RSVP configuration file.

This example:

- Runs the RSVP Agent on the stack selected using the standard resolver search order, because a TcpImage statement is not configured.
- Disables RSVP processing on interface 10.11.12.13, while enabling it for all other interfaces.
- Disables traffic control on interface 200.1.1.1. This means that no reservations will be made on this interface.
- Allows a maximum of 50 active RSVP flows per interface.

```
Interface 10.11.12.13 Disabled
{}
Interface 200.1.1.1 Enabled
{
TrafficControl Disabled
}
Interface Others Enabled
{}
Rsvp All Enabled
{
MaxFlows 50
}
```

Starting and stopping RSVP

RSVP can be started from the z/OS shell or as a started task.

The RSVP agent uses the following search order to locate the configuration file (highest priority is listed first):

- HFS file or MVS data set specified by the -c startup option. The syntax for an HFS file is '/dir/file', and the syntax for an MVS data set is "//'MVS.DATASET.NAME'".
- HFS file or MVS data set specified with the RSVPD_CONFIG_FILE enviroment variable.
- /etc/rsvpd.conf HFS file.
- 'hlq.RSVPD.CONF' MVS data set.

  Note: If this file is not present, RSVP is enabled on all network interfaces with default parameters.

When starting from the shell, note that the RSVP executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your path statement (in the profile) contains either /usr/sbin or /usr/lpp/tcpip/sbin.

Use the S RSVPD command on an MVS console or SDSF to start RSVP as a started task. A sample procedure is shipped in member EZARSVPP in SEZAINST.

RSVP can be stopped using the cancel command (C RSVPD) or using the kill command in the z/OS shell. The following kill command with the TERM signal will enable RSVP to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where *pid* is the RSVP process ID.

The RSVP process ID can be obtained using the following z/OS UNIX command:

```
ps  -A
```

It can also be obtained from the /tmp/rsvpd.pid.imagename file. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Service Level Agreement Performance Monitor MIB subagent

The SNMP Service Level Agreement (SLA) Performance Monitor MIB subagent provides information about defined service policies and performance data for applications which are mapped to those policies. Statistics are retrieved by this subagent and monitored for possible SLA performance deviations. Refer to RFC 2758 for more information about the SLAPM MIB.

Starting and stopping the SLA subagent

The SLA subagent can be started from the z/OS shell or as a started task.

When starting from the shell, note that the SLA subagent executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure your PATH statement (in the profile) contains either /usr/sbin or /usr/lpp/tcpip/sbin.

For example, the following command starts the SLA subagent with these characteristics:

```
pagtsnmp -d 1 -t 1800 -c special -P 5000
```

- To connect to the SNMP Agent, a community name of "special" and a port of 5000 are used.
- The debugging level is set to 1, meaning internal debugging messages are written to syslogd.

- The MIB table cache time is set to 30 minutes.

Use the S PAGTSNMP command on an MVS console or SDSF to start the SLA subagent as a started task. A sample procedure is shipped in member EZAPAGSN in SEZAINST.

The SLA Subagent can be stopped using the stop command (P PAGTSNMP) or using the kill command in the z/OS shell. The following kill command with the TERM signal will enable the SLA subagent to clean up resources properly before terminating itself:

```
kill -s TERM pid
```

where *pid* is where pid is the pagtsnmo process ID.

The pagtsnmp process ID can be obtained using the following z/OS UNIX command:

```
ps  -A
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the MIB objects supported by the SLA subagent.

Verification

To verify that policies are correctly defined and functioning properly, consider the following points:

- Are the policies installed in the TCP/IP stacks?
- Is the expected traffic mapping to the correct policies?
- Are the Sysplex Distributor policy functions working correctly?
- Does anything need to be tuned?

The following sections provide more details about these considerations.

Are the policies installed in the TCP/IP stacks?

Use the NETSTAT SLAP or onetstat -j command to display QoS policy statistics. This command only displays statistics for active QoS (installed) policies, so it can be used to verify the correct policies are installed, even if the statistics are all 0. Since the Policy Agent can install policies on multiple stacks, issue this command on each stack to verify the correct set of QoS policies is installed.

Is the expected traffic mapping to the correct QoS policies?

While connections are active, use the NETSTAT ALL or onetstat -A command to display details about the active connections. One piece of information displayed is the policy rule name. If this name is blank, then the traffic is not mapped to any active rule. Also, use the NETSTAT SLAP or onetstat -j command to display QoS policy statistics. The output shows the time that each policy was last mapped to traffic, and accumulated statistics for each policy. Monitor these values over time to verify that new traffic is mapping as expected.

Note: The values displayed by the NETSTAT SLAP and onetstat -j commands can wrap around to 0. If some of the values do not seem correct (for example, total out bytes less than total out bytes in profile), then wrapping has probably occurred.

Are the Sysplex Distributor policy functions working correctly?

To verify that the distributor is using the expected service levels when deciding how to distribute traffic to each DVIPA/Port target, use the Netstat VDPT DETAIL or onetstat -O DETAIL command on the distributing stack. The following QoS related information will be displayed for each DVIPA/Port target:

- WLM weight unmodified by QoS
- Modified WLM/QoS aggregate weight, identified by *DEFAULT*
- Modified WLM/QoS service level weights, identified by service level name

To verify that active connections distributed to DVIPA/Port targets are using the expected service level, use the Netstat VCRT DETAIL command on the distributing stack. This will display the following policy related information:

- PolicyRule: the policy rule that the distributor used in selecting the policy action for this connection.
- PolicyAction: the policy action that this connection is currently using. If PolicyAction is specified by *NONE*, then the distributor is using the *DEFAULT* fraction to distribute this connection.

Refer to *z/OS Communications Server: IP Diagnosis* for more information.

Does anything need to be tuned?

When poor performance (for example, low throughput, long response time, and so on) is experienced unexpectedly and rather consistently by a certain set of users or applications, or TRAPs are generated by the SLA Subagent, the problem might be in the way the QoS policy is defined for the corresponding set of users or applications. For example, the ToS/DS value might be set incorrectly to a lower QoS level than is intended, for example, medium or low priority instead of high priority. It is important to remember that given a fixed amount of network resources, changing some traffic demand from a lower to higher QoS level will mean that other traffic demands will be affected. Therefore, use care to ensure that in attempting to meet one set of QoS requirements, different or worse problems do not result.

Another cause for poor performance might be in the way the bandwidth allocation defined via the DiffServ token bucket parameters, or TCP maxrate or minrate, is not adequate to accommodate the traffic demand. Yet another possibility might be that either network or the server capacity is not adequate to handle the traffic demand. This is evident when a majority of users or applications do not have their QoS requirements met. When this happens, the network planning process must be revisited.

For more information, see “Using the SLA subagent to monitor policies” on page 588.

Using PASEARCH

Use the pasearch command to display policy details. This command displays both active (installed in the stack) and inactive policies. Various parameters can be specified to filter the results, for example to display only policies for certain stacks, only QoS policies, only policy names, or only a single policy specified by name. Refer to *z/OS Communications Server: IP System Administrator’s Commands* for the complete syntax and sample output for pasearch.

Using the SLA subagent to monitor policies

SLA subagent performance monitoring

The SLA Subagent provides information about service policies and performance data for applications mapped to those policies via two tables.

Note: The SLA subagent can be used to monitor Differentiated Services policies, and RSVP reservations if a corresponding RSVP policy is defined.

slapmPolicyRuleStatsTable
Provides information about defined service policies and aggregate performance data for mapped applications.

slapmSubcomponentTable
Provides information about individual TCP or UDP applications and application-specific performance data.

The SLA Subagent also supports performance monitoring via the slapmPRMonTable object. Entries are created in the monitor table to establish the desired criteria for monitoring. Two levels of monitoring are provided:

Aggregate
Monitoring is performed based on the aggregate of all TCP or UDP applications that are mapped to one or more service policies.

Subcomponent
Monitoring is performed based on a single TCP or UDP application.

Three types of monitoring are provided for measuring application performance:

MinRate
The current input/output rates of the applications are compared to threshold values established in the monitor table entry. If the current rates are less than the threshold, an SNMP trap is sent if traps are enabled.

MaxRate
The current input/output rates of the applications are compared to threshold values established in the monitor table entry. If the current rates are greater than the threshold, an SNMP trap is sent if traps are enabled.

MaxDelay
The current delay rates of the applications are calculated by using TCP round trip time (RTT). For aggregate monitoring, the RTT of all TCP applications are averaged. The delay rates are compared to threshold values established in the monitor table entry. If the current rates are greater than the threshold, an SNMP trap is sent if traps are enabled.

Note: MaxDelay monitoring is only available for TCP applications.

Refer to RFC 2758 for the SLAPM MIB in the sample file slapm.text in the /usr/lpp/tcpip/samples directory for more details about how to make the various monitoring calculations.

When SNMP traps are enabled, and a *not achieved* trap is sent as described above, a corresponding *okay* trap is sent when the traffic once again conforms to the boundaries established in the monitor table entry.

For example, you can establish MaxDelay monitoring and use a max delay low value of 50 and a max delay high value of 75. If the RTT of the applications rises

above 75, a *not achieved* trap is sent. If the RTT then drops below 50, an *okay* trap is sent to indicate the problem has been resolved. However, if the applications end before conforming to the established boundaries, an okay trap may not be sent naturally. For application level traps, the okay trap is never sent because the corresponding subcomponent table entry is deleted when the applications end, which also removes the application level monitoring.

For aggregate traps, when the applications end, MaxRate and MaxDelay okay traps will be sent, because a value of 0 for each of these should fall below the minimum values established in the monitor table entry. Conversely, for MinRate monitoring, an aggregate okay trap is not sent, because a value of 0 will never be greater than the maximum value established in the monitor table.

In addition to the traps used to measure application performance, two additional traps are used to monitor table administration:

Policy Deleted
Trap is sent when an entry is deleted from the slapmPolicyRuleStatsTable.

Monitor Deleted
Trap is sent when an entry is deleted from the slapPRMonTable.

Creating monitor table entries and enabling SNMP traps: Several MIB objects are used when establishing monitor table entries and for configuring whether and how often traps are sent. First, to establish monitor table entries, set the following MIB object variables.

Note: Because most of these objects have default values, you might be able to achieve the desired monitoring using only a subset of the objects.

slapmPRMonControl — Controls what levels and types of monitoring are in effect.

slapmPRMonInterval — Sets the interval for calculating input/output and delay rates and checks those values against the monitor table thresholds.

slapmPRMonMinRateLow, slapmPRMonMinRateHigh, slapmPRMonMaxRateLow, slapmPRMonMaxRateHigh, slapmPRMonMaxDelayLow, slapmPRMonMaxDelayHigh
Establishes the threshold values for MinRate, MaxRate, and MaxDelay monitoring. The min and max rates are in units of kilobits per second, and the max delay is in units of milliseconds.

slapmPRMonRowStatus — Controls the status of a monitor table entry, for instance whether or not the entry is active.

In addition, the following MIB objects are used to control the generation of traps:

slapmPolicyTrapEnable
Enables or suppresses generation of Policy Deleted and Monitor Deleted traps.

slapmPolicyTrapFilter
Establishes the number of times a given MinRate, MaxRate, or MaxDelay event must be encountered before a trap is generated.

slapmPolicyPurgeTime
Establishes a timeout value for Policy Deleted traps. After a service policy is deleted, the amount of time indicated by this object must expire before a Policy Deleted trap is generated.

slapmPRMonControl
Controls whether or not aggregate and/or subcomponent traps are enabled.

Creating the monitor table index: When you create monitor table entries, specify the appropriate index value. The index is composed of the following

- slapmPRMonOwnerIndex
- slapmPRMonSystemAddress
- slapmPRMonIndex

The OwnerIndex is expressed in the following format:

```
length.character.character...
```

where *character* is in ASCII decimal form.

For example, the value *u1* is expressed as 2.117.31. The SystemAddress value will always be 0. The Index value is the index into the slapmPolicyNameTable that maps to the policy name.

For example, assume the following service policy is defined:

```
PolicyAction                ElaineCat
     {
          PolicyScope             RSVP
          MaxRatePerFlow          640
          MaxTokenBucketPerFlow   440
          Maxdelay                1
     }
     PolicyRule                   ElainePol
     {
          ProtocolNumberRange     6
          SourcePortRange         8000
          PolicyActionReference   ElaineCat
     }
```

The SLA Subagent will create an entry in the slapmPolicyNameTable to represent the policy rule. The index value for this entry is arbitrary and assigned by the subagent. Corresponding entries in the other MIB tables, including the monitor table, contain the index value that maps to the entry in the name table.

To assist you in creating the index for the monitor table entries, note that the index value used in the slapmPolicyRuleStatsTable entries consists of the last two values used in the monitor table index, namely the SystemAddress and Index. Thus, you can walk through the policy statistics table using the following command:

```
osnmp -v walk slapmPolicyRuleStatsTable
```

Then, cut and paste the index value from the PolicyRuleStatsTable and add an OwnerIndex of your choosing at the beginning of the index.

For the above example, the complete index using an OwnerIndex of ″u1″ is:

```
2.117.31.0.3
|        | +--- name table index value (Index)
|        +----- no SystemAddress (SystemAddress)
+-------------- length + "u1" (OwnerIndex)
```

Monitor table examples:

Note: If you are going to change any of the monitor table object values for an existing table entry or row, you must take the row out of service to make the changes. To do this, set the value of slapmPRMonRowStatus to 2.

After your changes are made, set the row status to a value of 1 to put it back in service.

Two of the monitor table objects, monitor control and monitor status fields, are important in setting up the entries and understanding why traps are generated. Both of these fields have the SNMP data type BITS, which means they are bit strings, where bit 0 is the low order bit. Any combination of bits can be set into these objects.

Table 21 shows the meaning of the various bits:

Table 21. Monitor control and monitor status object bit values

slapmPRMonControl Object	**xx54 3210**
0 - monitor MinRate	0000 0001 = 01
1 - monitor MaxRate	0000 0010 = 02
2 - monitor MaxDelay	0000 0100 = 04
3 - enable aggregate traps	0000 1000 = 08
4 - enable subcomponent traps	0001 0000 = 10
5 - monitor subcomponent	0010 0000 = 20
slapmPRMonStatus Object	xx98 7654 3210
0 - slaMinInRateNotAchieved	0000 0000 0001 = 001
1 - slaMaxInRateExceeded	0000 0000 0010 = 002
2 - slaMaxDelayExceeded	0000 0000 0100 = 004
3 - slaMinOutRateNotAchieved	0000 0000 1000 = 008
4 - slaMaxOutRateExceeded	0000 0001 0000 = 010
5 - monitorMinInRateNotAchieved	0000 0010 0000 = 020
6 - monitorMaxInRateExceeded	0000 0100 0000 = 040
7 - monitorMaxDelayExceeded	0000 1000 0000 = 080
8 - monitorMinOutRateNotAchieved	0001 0000 0000 = 100
9 - monitorMaxOutRateExceeded	0010 0000 0000 = 200

The following examples show how to create monitor table entries to monitor at various levels and types. You can also create other combinations using the monitor control object.

This example assumes SNMP version 1 security and no SNMPD.CONF file.

First, enable traps, assuming Version 1 security and no SNMPD.CONF file. The snmptrap.dest file should contain the IP address and protocol of an entity to receive traps. In this example, use the osnmp command running in the background to receive traps.

```
/etc/snmptrap.dest   contains: 9.67.191.5    UDP
  /etc/pw.src          contains: public 0.0.0.0    0.0.0.0
```

```
  osnmp set slapmPolicyTrapEnable.0 1
  osnmp set slapmPolicyTrapFilter.0 1
  osnmp set slapmPolicyPurgeTime.0 60
  osnmp trap > /tmp/trap.output &
```

To monitor for MinRate at an aggregate level:

```
osnmp set slapmPRMonControl.index \'00000009\'h
           where  9 is aggregate monitor and trap for MinRate
                  1 is aggregate monitor for MinRate

  osnmp set slapmPRMonMinRateLow.index  l
  osnmp set slapmPRMonMinRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MinRate occurs, then look at the monitor status object in the trap, or:

```
  osnmp get slapmPRMonStatus.index
           should be 1 if inbound MinRate not achieved
                     8 if outbound MinRate not achieved
```

To monitor for MaxRate at an aggregate level:

```
osnmp set slapmPRMonControl.index \'0000000a\'h
           where  a is aggregate monitor and trap for MaxRate
                  2 is aggregate monitor for MaxRate

  osnmp set slapmPRMonMaxRateLow.index  l
  osnmp set slapmPRMonMaxRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 2 if inbound MaxRate not achieved
                    10 if outbound MaxRate not achieved
```

To monitor for MaxDelay at an aggregate level:

```
osnmp set slapmPRMonControl.index \'0000000c\'h
           where  c is aggregate monitor and trap for MaxDelay
                  4 is aggregate monitor for MaxDelay

  osnmp set slapmPRMonMaxDelayLow.index  l
  osnmp set slapmPRMonMaxDelayHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxDelay occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 4 if MaxDelay exceeded
```

To monitor for MinRate at an application level:

```
osnmp set slapmPRMonControl.index \'00000031\'h
           where 31 is subcomponent monitor and trap for MinRate
                 21 is subcomponent monitor for MinRate

  osnmp set slapmPRMonMinRateLow.index  l
```

```
  osnmp set slapmPRMonMinRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MinRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 20 if inbound MinRate not achieved
                    100 if outbound MinRate not achieved
```

To monitor for MaxRate at an application level:

```
osnmp set slapmPRMonControl.index \'00000032\'h
           where 32 is subcomponent monitor and trap for MaxRate
                 22 is subcomponent monitor for MaxRate

  osnmp set slapmPRMonMaxRateLow.index  l
  osnmp set slapmPRMonMaxRateHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxRate occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 40 if inbound MinRate not achieved
                    200 if outbound MinRate not achieved
```

To monitor for MaxDelay at an application level:

```
osnmp set slapmPRMonControl.index \'00000034\'h
           where 34 is subcomponent monitor and trap for MaxDelay
                 24 is subcomponent monitor for MaxDelay

  osnmp set slapmPRMonMaxDelayLow.index  l
  osnmp set slapmPRMonMaxDelayHigh.index h
           where l is the lower boundary and h is the upper boundary

  osnmp set slapmPRMonRowStatus.index 1
```

If the MaxDelay occurs, then look at the monitor status object in the trap, or:

```
osnmp get slapmPRMonStatus.index
           should be 80 if MaxDelay exceeded
```

Chapter 13. Intrusion Detection Services (IDS)

It is becoming increasingly important to not just protect systems from attacks but to detect patterns of usage that might indicate impending attacks. Many attacks follow a sequence of information gathering, unauthorized access to resources (information, applications, storage) and denial of service. It can be difficult, or at times, impossible to determine the originator of denial of service attacks. Correlating information gathering activities with access violation may help identify an intruder before they succeed.

Intrusion Detection Services provides support for:

- Scan detection and reporting
- Attack detection, reporting and prevention
- Traffic regulation for TCP connections and UDP receive queues

Each of these is described in detail below.

IDS policies are used to specify what events are to be detected under what circumstances and what action to take. All IDS policies support logging events to a specified message level in syslogd and/or the system console. Most IDS policies support discarding packets when a specified limit is reached. Most IDS policies support writing statistics records to the INFO message level of Syslogd on a specified time interval, optionally only if exceptional events have occurred. All IDS policies support tracing all or part of the triggering packet to an IDS specific CTRACE facility, SYSTCPIS. IDS assigns a correlator value to each event. Messages written to the system console and syslogd and records written to the IDS trace all use this correlator. A single detected event may involve multiple packets. The correlator value identifies which messages and packets are related to each other. Each IDS policy has additional attributes that are specified either in conditions or in the action.

Scan policies

Scans are recognized as the result of multiple information gathering events from a single source IP within a defined period of time. Scanning in and of itself is not harmful. However, many serious attacks, especially access violation attacks, are preceded by information gathering scans. Because scans by their nature must use reliable source IP addresses, they can be interesting events to monitor.

The IDS support defines a scanner as a source host that accesses multiple unique resources (ports or interfaces) over a specified period of time. The number of unique resources (Threshold) and the time period (Interval) can be specified via policy. Two categories of scans are supported:

- Fast scan
  - many resources rapidly accessed in a short time period (usually less than 5 minutes and program driven)
- Slow scan
  - different resources intermittently accessed over a longer period of time (many hours). This could be a scanner trying to avoid detection.

Sample scanners:

- Source host A has a program that loops through all low ports and tries to connect to each port on target host X (fast scan). Note: each port is considered a unique resource.
- Source host B manually does pings to each interface on target host X and then tries to access well-known ports on target host X (most likely a slow scan) . Note: each interface accessed by the ping is considered a unique resource and each port accessed is considered a unique resource.

Not a scanner:

- Source host C starts 20 connections to port 23 . Since these connections are to the same port, only one unique resource has been accessed. Therefore, host C is not considered a scanner.

Certain scans may not be detected by IDS:

- source host E issues pings to addresses 9.1.1.1 through 9.255.255.255. Since host X only collects data for the pings directed to X's interfaces, this is not detected by host X as a scan. Network IDS may detect this as wide scan.

Scan policy provides the ability to:

- Control the parameters that define a scan:
  - Fast scan time interval
  - Slow scan time interval
  - Fast scan threshold
  - Slow scan threshold
  - Exclude well known legitimate scanners via an exclusion list
  - Specify a sensitivity level by port or portrange (to reduce performance impacts)
  - Notify the installation of a detected scan via console message or syslogd message
  - Trace potential scan packets

The individual packets used in a scan can be categorized as normal, possibly suspicious or very suspicious. To control the performance impact and analysis load of scan monitoring, it will be useful to have a mechanism for adjusting our interest level in potential scan events. For information gathering we will provide sensitivity levels of High, Medium and Low to control recognizing countable events for normal, possibly suspicious and very suspicious packets.

The following table shows how the policy-specified sensitivity affects the counting of scan events. The event suspicion level is determined by the stack.

Sensitivity (from policy)	Normal event	Possibly suspicious event	Very suspicious event
Low			count
Medium		count	count
High	count	count	count

To help reduce or eliminate false positives, IDS will allow policy-specified source IP addresses, subnet masks, and (optionally) source port numbers to be excluded

from scan detection. For UDP and TCP port scans, scan detection can be limited to specified destination port ranges. The sensitivity (high, medium and low) may be specified by these port ranges.

Another way IDS will reduce false positives is by counting only unique events from a specific source IP address within a scan interval. An event is considered unique if the IP Protocol, Destination IP Address and Destination Port (UDP, TCP) or Type (ICMP) have not been seen before within this scan interval.

IDS scan policy supports a fast scan interval and threshold and a slow scan interval and threshold. A fast scan will be recognized if more than the fast scan threshold-specified unique events are received. A slow scan will be recognized if more than the slow scan threshold-specified unique events are received. Counting of scan events will be done on an internal interval no greater than half of the fast scan interval to avoid missing scans that occur within the fast scan interval but spread across two reporting intervals. Within an internal interval, once the number of unique events reaches the slow scan threshold, IDS knows that a scan has been detected and it is not necessary to continue to save information about additional related events in storage. This saves both storage and processing overhead. These events, however, will be traced if requested by policy via the ibm-idsTraceData attribute in the action.

Scan events come from the categories listed below. Any countable scan event will count against an origin source IP address. The total number of countable events from all categories is compared to the policy thresholds. When an origin source IP address has exceeded the policy-defined fast or slow threshold an event may be sent to the TRMD for logging to SYSLOG. Additionally, a console message may be issued and the packet may be logged to the IDS packet trace depending on the notification options in the action. When an origin source IP address has exceeded the policy-defined fast or slow threshold an event will be sent to the TRMD for logging to SYSLOG or console. Once a scan event is logged for a particular source IP address, no further scan events will be reportable within the specified fast interval. The intervals and thresholds for fast and slow scan are global, that is, only one definition of them is allowed across all event categories.

- ICMP Scans

  ICMP requests (Echo, Information, Timestamp, Subnet Mask) are used to map network topology. Any request sent to a subnet base or broadcast address will be treated as a very suspicious event. Echo Requests (ping) and Timestamp Requests are very common and will be treated as normal events when they do not include the IP Options for Record Packet Route or Record Timestamp. These options are intended to be used only with ICMP Echo Request packets. The stack ignores them on any other type of packet. The other types of requests are uncommon and will be treated as possibly suspicious events.

Request type	Destination address	Event classification
any	subnet base or broadcast	very suspicious
Information or Subnet Mask	single host	possibly suspicious
Echo with IP Option Record Route or Record Timestamp	single host	possibly suspicious
Echo or Timestamp	single host	normal

- UDP Port Scans

Because UDP is stateless, the stack is unable to differentiate between a client port and a server port. A scanner sending messages to many ephemeral ports looks very similar to a DNS server sending replies to many clients on ephemeral ports. TCP/IP configuration allows UDP ports to be RESERVED, therefore restricting a port so that it cannot be used. Any datagram received for a restricted port will be treated as a highly suspicious event. Datagrams received for unbound but unrestricted ports will be treated as possibly suspicious events and datagrams received for bound ports will be treated as normal events. Event generation can also be scoped to specific port ranges.

Socket state	Event	Event classification
Restricted (RESERVED to no one	recv any packet	very suspicious
Unbound, not restricted	recv any packet	possibly suspicious scanner or application temporarily down
Bound	recv any packet	normal

- TCP Port Scans

  Because TCP is a stateful protocol, there are many different events that may be classified as normal, possibly suspicious or highly suspicious. The identified conditions are listed in the table that follows. TCP/IP configuration allow TCP ports to be RESERVED, therefore restricting a port so that it cannot be used. Event generation can also be scoped to specific port ranges.

Socket state	Event	Event classification
Any state	recv unexpected flags (i.e. SYN+FIN)	very suspicious
Restricted (RESERVED to no one	recv any packet	very suspicious
Unbound, not restricted	recv any packet	possibly suspicious scanner or application temporarily down
Listen	recv standalone SYN	no event (classification deferred)
Half open connection	recv ACK	normal - connection handshake completed
Half open connection	recv RST	possibly suspicious peer covering tracks
Half open connection	final time out	very suspicious peer abandoned handshake
Any connected state	seq# out of window	normal perhaps duplicate packet
Any connected state	recv standalone SYN	normal perhaps peer reboot
Any connected state	final timeout	possibly suspicious peer abandoned connection

Attack policies

An attack can be a single packet designed to crash or hang a system. An attack can also consist of multiple packets designed to consume a limited resource causing a network, system or application to be unavailable to its intended users (i.e. denial of service). IDS attack policy allows you to turn on attack detection for one or

more categories of attacks independently of each other. In general, the types of actions that can be specified for an attack policy are event logging, statistics gathering, packet tracing and discarding of the attack packets.

Most attack checking is done for inbound packets destined for this stack. The IDS categories of attacks are:

- Malformed packets events

  There are numerous attacks designed to crash a system's protocol stack by providing incorrect or partial header information. These packets are always discarded when received regardless of IDS policy. The source IP address is rarely reliable for these attacks.

  You can use IDS policy to provide notification of malformed packet attacks.
- Inbound fragment restrictions

  Many attacks are the result of fragment overlays in the IP or transport header. This support allows you to protect your system against future attacks by detecting fragmentation within the first 256 bytes of a datagram.

  You can use IDS policy to provide notification of a packet that results from a datagram being fragmented in the first 256 bytes, as well as to discard the packet.
- IP protocol restrictions

  While there are 256 possible valid IP protocols, only a handful are in common usage today. This support allows you to protect your system against future attacks by prohibiting those protocols that you are not actively and intentionally using.

  You can use IDS policy to provide notification of a packet with a restricted IP protocol, as well as to discard the packet.
- IP option restrictions

  As with IP protocols, there are 256 possible IP options, with only a small number currently in common use. This support allows you to prevent misuse of options you are not intentionally using. Note that checking for restricted IP options is performed on all inbound packets, even those forwarded to another system.

  You can use IDS policy to provide notification of a packet with a restricted IP option, as well as to discard the packet.
- UDP perpetual echo

  Some UDP applications unconditionally respond to every datagram received. In some cases, such as Echo, CharGen or TimeOfDay, this is a useful network management or network diagnosis tool. In other cases it may be polite application behavior to send error messages in response to incorrectly formed requests. If a datagram is inserted into the network with one of these applications as the destination and another of these applications spoofed as the source, the two applications will respond to each other continually. Each inserted datagram will result in another perpetual echo conversation between them. This support allows you to define the application ports that exhibit this behavior.

  You can use IDS policy to provide notification of a perpetual echo packet, as well as to discard the packet.
- ICMP redirect restrictions

  ICMP redirect packets can be used to modify your routing tables. The IGNOREREDIRECT statement in the TCPIP profile disables ICMP Redirects. You can use IDS policy to provide notification of attempts to modify your routing tables in this manner.

You can also use IDS policy to disable ICMP Redirects. ICMP Redirect packets will be ignored or discarded if either IGNOREREDIRECT is specified in the TCPIP profile or if IDS policy is active for ICMP redirect attacks and the associated policy action requests that the packet be discarded (ibm-idsTypeActions:LIMIT).

- Outbound raw restrictions

  Most network attacks require the ability to craft packets that would not normally be built by a proper protocol stack implementation. This support allows you to detect and prevent many of these crafting attempts so that your system is not used as the source of attacks on other systems. As part of this checking, you can restrict the IP protocols allowed in an outbound RAW packet. It is recommended that you restrict the TCP protocol (6) on the outbound raw rule.

  You can use IDS policy to provide notification of an outbound raw packet that is considered an attack, as well as to discard the packet.

- TCP SYNflood Flood events

  One popular denial of service attack is to flood a public server with connection requests from incorrect or nonexistent source IP addresses. The intent is to use up the available slots for connection requests and thereby deny legitimate access from completing. z/OS CS provides protection from this attack regardless of IDS policy.

  You can use IDS policy to provide notification of an attack so that you may address the situation with your network administrators and service providers in a timely manner. Notification of a flood can include flood start and flood end event messages and tracing of the first 100 packets discarded due to the flood.

For each attack category (e.g. restricted IP protocol) the single highest priority rule is mapped at policy change.

One or more notification options can be specified in the action to provide the desired documentation of detected attacks.

For IDS attack policy the ibm-idsNotification attribute allows attack events to be logged to syslogd and/or the system console. Note that the console messages provide a subset of the information provided in the syslogd messages. For all attack categories except flood, a single packet triggers an event. To prevent message flooding to the system console, you can specify the maximum number of console messages to be logged per attack category within a 5 minute interval (ibm-idsMaxEventMessage). If you specify logging to the console in your IDS policy it is recommended that you specify a maximum event message; there is no default. To prevent message flooding to syslogd, a maximum of 100 event messages per attack category will be logged to syslogd within a 5 minute interval.

For IDS attack policy the statistics action provides a count of the number of attack events detected during the statistics interval. The count of attacks is kept separately for each category of attack (e.g. malformed) and a separate statistics record is generated for each. If you want to turn on statistics for attacks, it is recommended that you specify exception statistics (ibm-idsTypeActions:EXCEPTSTATS). With exception statistics, a statistics record will only be generated for the category of attack if the count of attacks is nonzero. If statistics is requested (ibm-idsTypeActions:STATISTICS) a record will be generated every statistics interval regardless of whether an attack has been detected during that interval or not.

For IDS attack policy the ibm-idsTraceData and ibm-idsTraceRecordSize attributes indicate if packets associated with attack events are to be traced. For all attack

categories except flood, a single packet triggers an event and the packet is traced. To prevent trace flooding, a maximum of 100 attack packets per attack category will be traced within a 5 minute interval. For the flood category, the first 100 hundred packets discarded during the flood will be traced.

It is applicable to all attack categories. However, Malformed and flood packets are always discarded regardless of this setting.

An action can be unique to a specific category of attack (e.g. malformed) or shared by one or more categories of attacks. If an action is shared, statistics data is still kept separately for each type of attack. Also, the maximum console message limit is enforced individually for each category of attack.

Traffic Regulation (TR) policies

IDS TR policies are used to limit memory resource consumption and queue delay time during peak loads.

TR TCP

IDS TR policies for TCP ports limit the total number of connections an application has active at one time. This can be used to limit the number of address spaces created by forking applications such as FTPD and otelnetd. A *fair share* algorithm is also provided based on the percentage of remaining available connections already held by a source IP address.

The percentage is applied against the number of available connections for the port. Therefore, as fewer connections become available, each host is allowed fewer new connections. The percentage is applied against the number of available connections, rather than the total number of connections allowed, in order to allow access to a larger number of different hosts when resources are low.

When a host requests a connection, the number of connections it currently holds for the port is compared to the percentage applied to the connections currently available for the port. If the number currently held is less than the percentage of currently available connections, the host is allowed to open an additional connection. If equal or greater, the host is not allowed to open further connections until more connections are freed up. All connection requesters for the port are regulated by this mechanism. If a host does not currently have any connections open on the port and unused connections are available, a host will always be allowed at least 1 connection. Multi-user source IP addresses may be allowed a larger number of connections by specifying a QoS policy with a higher number of connections (MaxConnections) than allowed by the TR policy. TR will honor the QoS differentiated services policy if the port is not in a *constrained state*. A QoS exception is made only when QoS differentiated service policy is applied for the specific source server port and specific outbound client destination IP address; if either of these attributes specify a range or are null, the QoS exception will not be made.

TR TCP generates a Constrained Event when a port reaches about 90% of its Connection Limit. An Unconstrained Event is generated when the port falls below about 88% of its limit. An IDS correlator is assigned for the duration of each constrained state. If tracing is requested in the policy, the first 100 packets that exceed the limit in each constrained state are traced along with the correlator. TR TCP also generates events for each connection allowed because of a QoS override

policy and for each connection denied for exceeding either the application's connection limit or the percent available limit.

TR UDP

Previously, control over UDP based applications consisted of application priority management and the TCP/IP profile parameter UDPQueueLimit ON | OFF. Inbound datagrams for bound UDP ports are accepted and queued until the queue limit is reached or buffer memory is exhausted. If UDPQueueLimit is set to OFF, any single bound port under a flood attack or with a stalled application could consume all available buffer storage. It is recommended that UDPQueueLimit always be set to ON. This limits the amount of storage that can be consumed by inbound datagrams for any single bound port. Sockets that use the Pascal API, have a limit of 160 KB in any number of datagrams. Sockets that use other APIs, have a limit of 2000 datagrams or 2880 KB.

IDS TR policies for UDP ports specify one of four abstract queue sizes for specified bound IP addresses and ports. The four abstract sizes are VERY_SHORT, SHORT, LONG and VERY_LONG. The actual queue sizes associated with these abstract values are internal values subject to change. Most UDP applications have timeout values based on human perceptions of responsiveness. These values tend to stay constant while system processing speeds and network delivery speeds continue to advance rapidly. This may require the physical sizes of these queues to change over time. The initial implementation uses the values of 16, 256, 2048 and 8192 (2**4, 2**8, 2**11, 2**13) for the number of datagrams and an average datagram size of 2 KB to calculate the byte sizes (32 KB, 512 KB, 4 MB, 16 MB). For performance reasons, sockets that use the Pascal API will only enforce the byte limit. Sockets that use other APIs will enforce both limits. Sockets without a policy specified for their port will use the existing UDPQueueLimit mechanism.

For applications that can process datagrams at a rate faster than the average arrival rate, the queue acts as a speed matching buffer that shifts temporary peak workloads into following valleys. The more that the application processing rate exceeds the average arrival rate and the larger the queue, the greater the variation in arrival rates that can be absorbed without losing work. Very fast applications with very bursty traffic patterns may benefit from LONG or VERY_LONG queue sizes.

For applications that consistently receive datagrams at a higher rate than they are able to process them, the queue acts to limit the effective arrival rate to the processing rate by discarding excess datagrams. In this case the queue size only influences the average wait time of datagrams in the queue and not the percentage of work lost. In fact, if the wait time gets too large, the peer application may have given up or retransmitted the datagram before it is processed. Slow applications with consistently high traffic rates may benefit from SHORT queue sizes.

In general, client side applications will tend to have lower system priority giving them lower datagram processing rates. They also tend to have much lower datagram arrival rates. Giving them SHORT or VERY_SHORT queue sizes may reduce the risk to system buffer storage under random port flood attacks with little impact on percentage of datagrams lost.

TR UDP generates a Constrained Event when a port reaches about 90% of its Queue Limit. An Unconstrained Event is generated when the port falls below about 88% of its limit. An IDS correlator is assigned for the duration of each constrained state. If tracing is requested in the policy, the first 100 packets that exceed the limit in each constrained state are traced along with the correlator.

Defining TR TCP policies using the Policy Agent

The TCP Traffic Management policy in IBM Communications Server for OS/390 V2R10 was defined as part of the QoS policy with a PolicyScope of TR. It could only be defined in the Policy Agent configuration file; the LDAP server was not supported. In z/OS CS, the TR function was incorporated into the IDS function. The full range of IDS policy is only available in LDAP. For upward compatibility, TR policy that existed previously will continue to be supported in the Policy Agent configuration file and will work unchanged, with the exception that it will be displayed differently by the pasearch command (as an IDS policy instead of a QoS policy with policy scope TR). However, if you want to use any of the expanded Intrusion Detection Services Traffic Regulation support, the expanded functions are only available through LDAP policy. You can use a combination of the LDAP policy for the new functions and keep the TCP traffic regulation policy in the configuration file, although this is not recommended.

Defining IDS policies using LDAP

IDS policies are stored in a server that supports LDAP, processed by Policy Agent and installed into a z/OS CS TCP/IP stack. You should be familiar with the information in Chapter 11, “Policy-Based Networking” on page 539 on running Policy Agent and LDAP before creating IDS policies.

A conservative approach to defining IDS policy will avoid unexpected application outages and excessive rule processing. The following examples describe policies provided in the sample files shipped with the system. (Refer to “Policy sample files” on page 541).

IDS policy definition considerations

IDS policies can be defined with different ibm-idsConditionType values. Each IDS policy must define exactly one ibm-idsConditionType. Specification of other additional conditions beyond those listed below will cause the rule to not be found. The supported types are:

SCAN_GLOBAL

This policy is searched by ibm-idsConditionType only. The single highest priority SCAN_GLOBAL rule is mapped at policy change and cached. The action defines the FastScan and SlowScan parameters as well as reporting and tracing actions to take when a scan is detected. The Limit and Statistics actions are ignored.

SCAN_EVENT

These policies are searched by ibm-idsConditionType and a protocol condition of ICMP, TCP or UDP. For protocols TCP and UDP the policy search includes local destination port and bound IP address as well. For ICMP, the single highest priority SCAN_EVENT rule is mapped at policy change and cached. The TCP and UDP rules are mapped when a potentially countable event occurs. If the event is associated with a bound socket, the rule is cached. The actions associated with these rules define the sensitivity level to use for counting events towards the scan thresholds and source exclusion list to use for the mapped events. Packet tracing occurs if the action associated with the SCAN_GLOBAL rule activates tracing and the sensitivity in the action associated with the SCAN_EVENT rule sensitivity indicates the event is countable.

ATTACK

There are several Attack Types. The conditions supported on each are

defined below. For each attack type, the single highest priority rule is mapped at policy change and cached. The reporting, tracing and statistics actions are supported for all attack types. Other supported actions are defined below. The supported attack types are:

MALFORMED_PACKET
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. The LIMIT action is ignored; malformed packets are always discarded by the stack, with or without policy.

FLOOD
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. The LIMIT action is ignored; excessive half-open TCP connections are always discarded by the stack, with or without policy.

ICMP_REDIRECT
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. ICMP redirect packets are discarded if either this policy specifies action LIMIT or the TCPIP PROFILE specifies IGNOREREDIRECT.

IP_FRAGMENT
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. If this policy specifies action LIMIT, datagrams that are fragmented within the first 256 bytes are discarded.

RESTRICTED_IP_OPTIONS
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. This attack type condition is expected to be ANDed with a list of conditions defining the IP options to disallow. If no IP options are specified, the rule will not accomplish anything. IP option 0 (end of list) and 1 (NO-OP) may not be disallowed and are ignored if specified. If this policy specifies action LIMIT, packets containing a disallowed option are discarded.

RESTRICTED_IP_PROTOCOL
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. This attack type condition is expected to be ANDed with a list of conditions defining the IP protocols to disallow. If no IP protocols are specified, the rule will not accomplish anything. IP protocols 1 (ICMP), 6 (TCP), and 17 (UDP) may not be disallowed and are ignored if specified. If this policy specifies action LIMIT, packets containing a disallowed protocol are discarded.

OUTBOUND_RAW
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. This attack type condition may optionally be ANDed with a list of conditions defining the IP protocols to disallow. If this policy specifies action LIMIT, any packet written to a RAW socket that has a source IP address not in the stack's home list, that is fragmented by the application, that specifies one of the ICMP reply types or that specifies a disallowed protocol are discarded.

PERPETUAL_ECHO
This policy is searched by ibm-idsConditionType and ibm-idsAttackType only. This attack type condition must be specified in a complex rule using CNF and multiple condition levels. The attack type condition is at one of the condition levels. There must

be a list of conditions defining the Local Port list at a second level. There must be a list of conditions defining the Remote Port list at a third level. Each of the port lists is limited by the stack to the first 20 ports specified. The negated flag is ignored by the stack on Port list conditions. Destination port is always checked against Local Port list. Source port is checked against the appropriate port list based on whether the source IP address is in the stack's home list. If this policy specifies action LIMIT, UDP packets with both ports in checked port lists are discarded.

TR These policies may optionally be ANDed with any combination of conditions defining protocol (TCP or UDP), local destination port or local destination IP address. TCP rules are mapped when a local application does a listen on a socket or when an inbound connection handshake completes. UDP rules are mapped when an inbound packet arrives at a local bound socket. UDP TR policy supersedes the TCPIP PROFILE setting of UDPQUEUELIMIT for covered ports. Mapped rules are cached and associated with the bound socket.

For TCP, the action defines the total number of allowed connections, the percentage of remaining available connections any single source IP may acquire and whether these limits are applied globally across all applications using this port number or applied individually to each application using an instance of this port number. For UDP, the action defines which of the four available queue sizes is applied to each application using this port number. TR actions define the reporting, statistics and tracing actions for covered ports. If the policy specifies action LIMIT, connections or packets that exceed the limits are discarded.

Notes:

1. For TCP, a total connection limit or percentage available limit of zero, with an action of LIMIT effectively quiesces the application.
2. For TCP, ibm-idsLocalHostIPAddress cannot be specified in any conditions if ibm-idsTRtcpLimitScope PORT specified.
3. For UDP, a policy for a port without an action of LIMIT effectively makes the application unlimited.
4. Each IDS TR action must specify at least one ibm-idsTypeActions.

IDS scan policy example

The goal of scan policy is to detect all scanners with potentially malicious intent while avoiding large numbers of false positives. You can make this process more efficient by reserving all unused low ports in the TCPIP profile. This will allow you to use the low sensitivity setting on scans for these ports. As you investigate the scans detected, you will initially find your own network management tools. These can be explicitly excluded. If you include UDP ephemeral ports in a high sensitivity policy, you will discover that your DNS servers show up as scanners. You can explicitly exclude these as well.

The following scan rules are defined:

- Scan Global

  Defines a global set of parameters for detecting scans, and also defines reporting parameters for scan events.

  – A Fast Scan is defined as 5 unique events in 2 minutes from a single source IP address.

  – A Slow Scan is defined as 10 unique events in 480 minutes (8 hours).

  - The first 200 bytes of the packet associated with each countable event will be traced.
  - When a scan is detected an event will be written to syslog warning level, along with a detailed list of all the unique events included in the scan.
  - No message will be written to the console.
  - Statistics records will not be written to syslog.
- Scan Event Low

  Defines a set of traffic for which low sensitivity scan detection will be performed. Inbound traffic to all TCP and UDP ports between 1 and 1023 will be monitored. It is recommended that unused low ports be RESERVED in the TCPIP Profile.
- Scan Event Medium

  Defines a set of traffic for which medium sensitivity scan detection will be performed. ICMP inbound traffic will be monitored.

```
dn:cn=scanglobal-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:scanglobal-rule
ibm-policyRuleName:ScanGlobal-rule
ibm-policyRuleConditionListType:2
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS global scan rule

dn:cn=condassoc1, cn=scanglobal-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsScanConditionAuxClass
cn:condassoc1
ibm-policyConditionName:ScanGlobal-condition
ibm-idsConditionType:SCAN_GLOBAL
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific condition

dn:cn=actassoc1, cn=scanglobal-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
objectclass:ibm-idsActionAuxClass
objectclass:ibm-idsScanActionAuxClass
objectclass:ibm-idsNotificationAuxClass
cn:actassoc1
ibm-policyActionName:ScanGlobal-action
ibm-idsActionType:SCAN_GLOBAL
ibm-policyActionOrder:1
ibm-idsTypeActions:LOG
ibm-idsNotification:SYSLOG
ibm-idsNotification:SYSLOGDETAIL
ibm-idsLoggingLevel:4
ibm-idsTraceData:RECORDSIZE
ibm-idsTraceRecordSize:200
ibm-idsFSInterval:2
ibm-idsFSThreshold:5
ibm-idsSSInterval:480
ibm-idsSSThreshold:10
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific action - Fast scan = 5 in 2 minutes, Slow scan = 10 in 8 hours

dn:cn=scaneventlow-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:scaneventlow-rule
ibm-policyRuleName:ScanEventLow-rule
ibm-policyRuleConditionListType:2
ibm-policyRuleEnabled:1
```

```
ibm-policyRulePriority:2
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:IDS scan event rule for low sensitivity on TCP and UDP Low Ports

dn:cn=condassoc2, cn=scaneventlow-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc2
ibm-policyConditionName:ScanEventLow-condition2
ibm-policyConditionDN:cn=ScanTcpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyConditionGroupNumber:2
ibm-policyConditionNegated:FALSE
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable CNF condition level 2 - scan TCP low ports

dn:cn=condassoc3, cn=scaneventlow-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc3
ibm-policyConditionName:ScanEventLow-condition3
ibm-policyConditionDN:cn=ScanUdpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyConditionGroupNumber:2
ibm-policyConditionNegated:FALSE
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable CNF condition level 2 - scan UDP low ports

dn:cn=actassoc1, cn=scaneventlow-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
objectclass:ibm-idsActionAuxClass
objectclass:ibm-idsScanSensitivityActionAuxClass
objectclass:ibm-idsScanExclusionActionAuxClass
cn:actassoc1
ibm-policyActionName:ScanEventLow-action
ibm-idsActionType:SCAN_EVENT
ibm-policyActionOrder:1
ibm-idsSensitivity:LOW
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific action - low sensitivity

dn:cn=scaneventmedium-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:scaneventmedium-rule
ibm-policyRuleName:ScanEventMedium-rule
ibm-policyRuleConditionListType:2
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:IDS scan event rule for medium sensitivity on ICMP

dn:cn=condassoc1, cn=scaneventmedium-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsScanEventConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:condassoc1
ibm-policyConditionName:ScanEventMedium-condition
ibm-idsConditionType:SCAN_EVENT
ibm-policyConditionGroupNumber:2
ibm-policyConditionNegated:FALSE
ibm-idsProtocolRange:1
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific condition - ICMP protocol

dn:cn=actassoc1, cn=scaneventmedium-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
objectclass:ibm-idsActionAuxClass
```

```
objectclass:ibm-idsScanSensitivityActionAuxClass
cn:actassoc1
ibm-policyActionName:ScanEventMedium-action
ibm-idsActionType:SCAN_EVENT
ibm-policyActionOrder:1
ibm-idsSensitivity:MEDIUM
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific action - medium sensitivity

dn:cn=ScanTcpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
ibm-idsConditionType:SCAN_EVENT
objectclass:ibm-idsScanEventConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:ScanTcpLowPorts
ibm-policyConditionName:ScanTcpLowPorts-condition
ibm-idsProtocolRange:6
ibm-idsLocalPortRange:1-1023
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS Scan TCP Low Ports condition

dn:cn=ScanUdpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
ibm-idsConditionType:SCAN_EVENT
objectclass:ibm-idsScanEventConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:ScabUdpLowPorts
ibm-policyConditionName:ScanUdpLowPorts-condition
ibm-idsProtocolRange:17
ibm-idsLocalPortRange:1-1023
ibm-policyKeywords:Scan
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS Scan UDP Low Ports condition
```

IDS attack policy examples

The goal of attack policy is to help protect your system from both known and unknown attacks and to give you timely notification when attacks do occur. Malformed packet policy covers many known attacks designed to cause system *crashes*. These packets are always discarded and rarely have legitimate source address information. Many malformed packet attacks use fragmentation to overlay header fields. The IDS fragment restriction policy is intended to protect you from unknown attacks of this type by disallowing fragmentation in the first 256 bytes of any datagram. Unless you know you need ICMP redirect, you should disallow it with policy. There are several types of flood attacks. IDS can identify TCP SYN floods. IDS policy should be used to notify you when a flood occurs. You will need to work with your network administrators and service providers to track the flood backwards, one physical hop at a time, to locate the source(s).

The IP protocol restrictions and IP option restrictions provide additional protection against future unknown attacks. The philosophy behind them is to disallow anything that you do not have a known reason to allow. The outbound raw policy is intended to help you detect someone using your system as the base for an attack. It looks for several behaviors associated with *spoofed* packets.

Attack rules define the set of conditions that define what constitutes an attack for a given attack type. The highest priority rule of each attack type is used. The action associated with an attack rule defines reporting and logging options for a detected attack.

The following types of attack rules are defined:

- Malformed Packet: Various types of known attacks based on malformed packets.
- Flood: TCP SYN flood attacks.
- ICMP Redirect: Disallows ICMP redirect receipts.
- IP Fragment: Disallows Fragmentation within first 256 bytes of datagrams.
- IP Protocol: Defines disallowable IP protocols.
  - Uses complex conditions to disallow everything except ICMP, TCP and UDP.
  - Uses DNF (condition list type 1) to evaluate complex conditions.
  - Conditions in the same group are ANDed together, groups are ORed.
- Outbound Raw Restrictions: Validity checking for outbound packets using RAW sockets.
  - Uses complex conditions to disallow everything except ICMP, UDP, IGMP and OSPFIGP.
- Several reusable Protocol conditions are defined that can be shared between the IP Protocol Restriction rule and the Outbound Raw rule.
- A single reusable attack action is defined and shared among all the attack rules.
  - Events are written to syslog ALERT level.
  - Events are not written to the system console.
  - The first 200 bytes of packets associated with an attack are traced.
  - Statistics are evaluated every 60 minutes and only written if an attack occurred.
  - Limit was not specified, so packets associated with IP Protocol Restrictions, IP Fragment Restriction and Outbound Raw Restrictions will not be deleted.

```
dn:cn=attackMalformed-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:attackMalformed-rule
ibm-policyRuleName:AttackMalformed-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS attack rule for Malformed Packets

dn:cn=condassoc1, cn=attackMalformed-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsAttackConditionAuxClass
cn:condassoc1
ibm-policyConditionName:attackMalformed-condition
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-idsConditionType:ATTACK
ibm-idsAttackType:MALFORMED_PACKET
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific condition - attack type

dn:cn=actassoc1, cn=attackMalformed-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:attackMalformed-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=attackact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - attack action 1

dn:cn=attackFlood-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
```

```
objectclass:ibm-policyRule
cn:attackFlood-rule
ibm-policyRuleName:AttackFlood-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS attack rule for Floods

dn:cn=condassoc1, cn=attackFlood-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsAttackConditionAuxClass
cn:condassoc1
ibm-policyConditionName:attackFlood-condition
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-idsConditionType:ATTACK
ibm-idsAttackType:FLOOD
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific condition - attack type

dn:cn=actassoc1, cn=attackFlood-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:attackFlood-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=attackact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - attack action 1

dn:cn=attackICMPRedirect-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:attackICMPRedirect-rule
ibm-policyRuleName:AttackICMPRedirect-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS attack rule for ICMP Redirect

dn:cn=condassoc1, cn=attackICMPRedirect-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsAttackConditionAuxClass
cn:condassoc1
ibm-policyConditionName:attackICMPRedirect-condition
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-idsConditionType:ATTACK
ibm-idsAttackType:ICMP_REDIRECT
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific condition - attack type

dn:cn=actassoc1, cn=attackICMPRedirect-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:attackICMPRedirect-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=attackact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - attack action 1

dn:cn=attackIpFragment-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
```

```
objectclass:ibm-policyRule
cn:attackIpFragment-rule
ibm-policyRuleName:AttackIpFragment-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS attack rule for IP fragment restriction

dn:cn=condassoc1, cn=attackIpFragment-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsAttackConditionAuxClass
cn:condassoc1
ibm-policyConditionName:attackIpFragment-condition
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-idsConditionType:ATTACK
ibm-idsAttackType:IP_FRAGMENT
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Rule-specific condition - attack type

dn:cn=actassoc1, cn=attackIpFragment-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:attackIpFragment-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=attackact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - attack action 1

dn:cn=attackIpProt-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:attackIpProt-rule
ibm-policyRuleName:AttackIPprot-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS attack rule for restricted protocol

dn:cn=condassoc1, cn=attackIpProt-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
ibm-policyConditionName:AttackIPprot-condition1
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=attackIpProtcond1, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents first reusable DNF condition at level 1

dn:cn=condassoc1a, cn=attackIpProt-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1a
ibm-policyConditionName:AttackIPprot-condition1a
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:TRUE
ibm-policyConditionDN:cn=IpProtICMP, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents second reusable DNF condition at level 1 (negated) allow ICMP

dn:cn=condassoc1b, cn=attackIpProt-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1b
```

```
ibm-policyConditionName:AttackIPprot-condition1b
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:TRUE
ibm-policyConditionDN:cn=IpProtTCP, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents third reusable DNF condition at level 1 (negated) allow TCP

dn:cn=condassoc1c, cn=attackIpProt-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1c
ibm-policyConditionName:AttackIPprot-condition1c
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:TRUE
ibm-policyConditionDN:cn=IpProtUDP, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents fourth reusable DNF condition at level 1 (negated) allow UDP

dn:cn=actassoc1, cn=attackIpProt-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:AttackIPprot-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=attackact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - attack action 1

dn:cn=attackOutboundRaw-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:attackOutboundRaw-rule
ibm-policyRuleName:AttackOutboundRaw-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRulePriority:2
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS attack rule for Outbound Raw restrictions

dn:cn=condassoc1, cn=attackOutboundRaw-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
ibm-policyConditionName:AttackOutboundRaw-condition1
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=attackOutboundRawcond1, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents first reusable DNF condition at level 1

dn:cn=condassoc1a, cn=attackOutboundRaw-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1a
ibm-policyConditionName:AttackOutboundRaw-condition1a
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:TRUE
ibm-policyConditionDN:cn=IpProtICMP, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents second reusable DNF condition at level 1 (negated) allow ICMP

dn:cn=condassoc1b, cn=attackOutboundRaw-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1b
ibm-policyConditionName:AttackOutboundRaw-condition1b
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:TRUE
ibm-policyConditionDN:cn=IpProtUDP, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
```

```
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents third reusable DNF condition at level 1 (negated) allow UDP

dn:cn=condassoc1c, cn=attackOutboundRaw-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1c
ibm-policyConditionName:AttackOutboundRaw-condition1c
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:TRUE
ibm-policyConditionDN:cn=IpProtIGMP, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents fourth reusable DNF condition at level 1 (negated) allow IGMP

dn:cn=condassoc1d, cn=attackOutboundRaw-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1d
ibm-policyConditionName:AttackOutboundRaw-condition1d
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:TRUE
ibm-policyConditionDN:cn=IpProtOSPFIGP, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents fifth reusable DNF condition at level 1 (negated) allow OSPFIGP

dn:cn=actassoc1, cn=attackOutboundRaw-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:AttackOutboundRaw-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=attackact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - attack action 1

dn: cn=attackact1, cn=IDSact, cn=repository, o=IBM, c=US
objectclass:ibm-policyActionInstance
objectclass:ibm-idsActionAuxClass
objectclass:ibm-idsNotificationAuxClass
objectclass:ibm-idsAttackActionsAuxClass
cn:attackact1
ibm-policyActionName:AttackLog-action
ibm-idsActionType:ATTACK
ibm-idsTypeActions:LOG
ibm-idsNotification:SYSLOG
ibm-idsLoggingLevel:1
ibm-idsTypeActions:EXCEPTSTATS
ibm-idsStatInterval:60
ibm-idsTraceData:RECORDSIZE
ibm-idsTraceRecordSize:200
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:IDS common attack action - LOG(SYSLOG(1) NOCONSOLE) NOLIMIT
description:IDS common attack action - EXECPTSTATS(60) TRACE(200)

dn:cn=attackIpProtcond1, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsAttackConditionAuxClass
cn:attackIpProtcond1
ibm-policyConditionName:AttackIPprot-condition1
ibm-idsConditionType:ATTACK
ibm-idsAttackType:RESTRICTED_IP_PROTOCOL
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS attack condition 1 for restricted IP protocol

dn:cn=attackOutboundRawcond1, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
```

```
objectclass:ibm-idsAttackConditionAuxClass
cn:attackOutboundRawcond1
ibm-policyConditionName:AttackOutboundRaw-condition1
ibm-idsConditionType:ATTACK
ibm-idsAttackType:OUTBOUND_RAW
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS attack condition 1 for Outbound Raw restrictions

dn:cn=IpProtICMP, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:IpProtICMP
ibm-policyConditionName:IpProtICMP
ibm-idsConditionType:ATTACK
ibm-idsProtocolRange:1
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS condition for IP protocol ICMP

dn:cn=IpProtIGMP, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:IpProtIGMP
ibm-policyConditionName:IpProtIGMP
ibm-idsConditionType:ATTACK
ibm-idsProtocolRange:2
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS condition for IP protocol IGMP

dn:cn=IpProtTCP, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:IpProtTCP
ibm-policyConditionName:IpProtTCP
ibm-idsConditionType:ATTACK
ibm-idsProtocolRange:6
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS condition for IP protocol TCP

dn:cn=IpProtUDP, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:IpProtUDP
ibm-policyConditionName:IpProtUDP
ibm-idsConditionType:ATTACK
ibm-idsProtocolRange:17
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS condition for IP protocol UDP

dn:cn=IpProtOSPFIGP, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:IpProtOSPFIGP
ibm-policyConditionName:IpProtOSPFIGP
ibm-idsConditionType:ATTACK
ibm-idsProtocolRange:89
ibm-policyKeywords:Attack
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS condition for IP protocol OSPFIGP
```

Traffic Regulation (TR) policy examples

The goal of TR policy is to protect your system from usage spikes. A phased approach to determine the correct policy for your system is recommended.

To gather baseline statistics, an installation will first need to run in Statistics mode, with the traffic regulation daemon (TRMD) running. In statistics mode, the following information is provided for the port on a policy defined interval:

- Total number of connections requested during the interval
- Total number of connections closed during the interval
- The IP address of the host that requested a connection during the interval and held the highest number of concurrent connections during the interval, and the highest number of concurrent connections held by this IP address
- A suggested value for TotalConnections based on this interval
- A suggested value for Percentage based on this interval

While the baseline statistics records provide suggested policy values for the interval, the installation should evaluate data from multiple intervals. The values suggested are those that would avoid denying any of the connections in the interval. Choose lower values if the interval represents a workload larger than you want to allow.

After the installation determines the policy values to use, try running with the Log action specified. Specifying the Log action, without the Limit action, basically tests out the policy. The connections that would have been denied (if the Limit action was specified) are logged, but the connection is allowed to occur. After the installation is satisfied with the experimental policy, the policy action can be set to Limit.

The following traffic regulation TCP rules are defined:

- TR TCP: Defines TCP baseline STATISTICS gathering for the low port range.
  - This rule provides statistics reports to determine normal traffic patterns for several applications.
- TR TCP WEB: Defines Application limits and Host percentage limits for a single application.
  - This rule enforces set limits.
  - The rule has a higher priority than the TR TCP rule.
  - The rule is limited to a single server application that is bound to a specific IP address.

```
dn:cn=trtcp-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:trtcp-rule
ibm-policyRuleName:TRtcp-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRuleConditionListDN:cn=condassoc1,cn=trtcp-rule,cn=IDS,cn=starter,
     ou=policy,o=IBM,c=US
ibm-policyRuleActionListDN:cn=actassoc1,cn=trtcp-rule,cn=IDS,cn=starter,
     ou=policy,o=IBM,c=US
ibm-policyRulePriority:2
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS TR TCP rule

dn:cn=condassoc1, cn=trtcp-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
```

```
ibm-policyConditionName:TRtcp-condition1
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=TrTcpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable condition - TR TCP low ports

dn:cn=actassoc1, cn=trtcp-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:TRtcp-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=trtcpact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - TR TCP action 1

dn:cn=trtcpWeb-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:trtcpWeb-rule
ibm-policyRuleName:trtcpWeb-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRuleConditionListDN:cn=condassoc1,cn=trtcpWeb-rule,cn=IDS,cn=advanced,
     ou=policy, o=IBM,c=US
ibm-policyRuleActionListDN:cn=actassoc1,cn=trtcpWeb-rule,cn=IDS,cn=advanced,
     ou=policy,o=IBM,c=US
ibm-policyRuleValidityPeriodList:cn=period1, cn=time, cn=repository, o=IBM, c=US
ibm-policyRulePriority:7
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS TR TCP rule with limit

dn:cn=condassoc1, cn=trtcpWeb-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
ibm-policyConditionName:TRtcpWeb-condition1
ibm-policyConditionGroupNumber:1
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=TrTcpWebPort, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable condition - TR TCP web port

dn:cn=actassoc1, cn=trtcpWeb-rule, cn=IDS, cn=advanced, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:TRtcpWeb-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=trtcpact2, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - TR TCP action 2

dn: cn=trtcpact1, cn=IDSact, cn=repository, o=IBM, c=US
objectclass:ibm-policyActionInstance
objectclass:ibm-idsActionAuxClass
objectclass:ibm-idsNotificationAuxClass
objectclass:ibm-idsTRtcpActionAuxClass
cn:trtcpact1
ibm-policyActionName:TRtcpLog-action
```

```
ibm-idsActionType:TR
ibm-idsTypeActions:LOG
ibm-idsNotification:SYSLOG
ibm-idsLoggingLevel:4
ibm-idsTypeActions:STATISTICS
ibm-idsStatInterval:60
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:IDS TR TCP action TCP(64K,100%) LOG(SYSLOG(4) NOCONSOLE) NOLIMIT
description:TRACE(HEADER) STATISTICS(60)

dn:cn=trtcpact2, cn=IDSact, cn=repository, o=IBM, c=US
objectclass:ibm-policyActionInstance
objectclass:ibm-idsActionAuxClass
objectclass:ibm-idsNotificationAuxClass
objectclass:ibm-idsTRtcpActionAuxClass
cn:trtcpact2
ibm-policyActionName:TRtcpLimit-action
ibm-idsActionType:TR
ibm-idsTypeActions:LIMIT
ibm-idsTypeActions:LOG
ibm-idsNotification:SYSLOG
ibm-idsLoggingLevel:4
ibm-idsTypeActions:EXCEPTSTATS
ibm-idsStatInterval:60
ibm-idsTRtcpTotalConnections:1000
ibm-idsTRtcpPercentage:10
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:IDS TR TCP action TCP(1K,10%) LOG(SYSLOG(4) NOCONSOLE) LIMIT
description:TRACE(HEADER) EXCEPTSTATS(60)

dn:cn=TrTcpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
ibm-idsConditionType:TR
objectclass:ibm-idsTrafficRegulationConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:TrTcpLowPorts
ibm-policyConditionName:TrTcpLowPorts-condition
ibm-idsProtocolRange:6
ibm-idsLocalPortRange:1-1023
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS TR TCP Low Ports condition

dn:cn=TrTcpWebPort, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
ibm-idsConditionType:TR
objectclass:ibm-idsTrafficRegulationConditionAuxClass
objectclass:ibm-idsHostConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:TrTcpWebPort
ibm-policyConditionName:TrTcpWebPort-condition
ibm-idsProtocolRange:6
ibm-idsLocalPortRange:80
ibm-idsLocalHostIPAddress:3-10.14.243.87
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS TR TCP Web Port condition
```

The following traffic regulation UDP rule is defined:

- TR UDP: Defines UDP queue size for the low port range.
  - This rule provides statistics reports to determine normal traffic patterns for several applications.

```
dn:cn=trudp-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRule
cn:trudp-rule
ibm-policyRuleName:TRudp-rule
ibm-policyRuleConditionListType:1
ibm-policyRuleEnabled:1
ibm-policyRuleConditionListDN:cn=condassoc1,cn=trudp-rule,cn=IDS,cn=starter,
     ou=policy,o=IBM,c=US
ibm-policyRuleActionListDN:cn=actassoc1,cn=trudp-rule,cn=IDS,cn=starter,
     ou=policy,o=IBM,c=US
ibm-policyRulePriority:2
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Example of IDS TR UDP rule

dn:cn=condassoc1, cn=trudp-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleConditionAssociation
cn:condassoc1
ibm-policyConditionName:TRudp-condition1
ibm-policyConditionGroupNumber:7
ibm-policyConditionNegated:FALSE
ibm-policyConditionDN:cn=TrUdpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable condition - TR UDP low ports

dn:cn=actassoc1, cn=trudp-rule, cn=IDS, cn=starter, ou=policy, o=IBM, c=US
objectclass:ibm-policyRuleActionAssociation
cn:actassoc1
ibm-policyActionName:TRudp-action
ibm-policyActionOrder:1
ibm-policyActionDN:cn=trudpact1, cn=IDSact, cn=repository, o=IBM, c=US
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Represents reusable action - TR UDP action 1

dn: cn=trudpact1, cn=IDSact, cn=repository, o=IBM, c=US
objectclass:ibm-policyActionInstance
objectclass:ibm-idsActionAuxClass
objectclass:ibm-idsNotificationAuxClass
objectclass:ibm-idsTRudpActionAuxClass
cn:trudpact1
ibm-policyActionName:TRudpLog-action
ibm-idsActionType:TR
ibm-idsTypeActions:LOG
ibm-idsNotification:SYSLOG
ibm-idsLoggingLevel:4
ibm-idsTypeActions:STATISTICS
ibm-idsStatInterval:60
ibm-idsTRudpQueueSize:LONG
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:IDS TR UDP action UDPQ(LONG) LOG(SYSLOG(4) NOCONSOLE) NOLIMIT
description:TRACE(HEADER) STATISTICS(60)

dn:cn=TrUdpLowPorts, cn=IDScond, cn=repository, o=IBM, c=US
objectclass:ibm-policyConditionInstance
objectclass:ibm-idsConditionAuxClass
ibm-idsConditionType:TR
```

```
objectclass:ibm-idsTrafficRegulationConditionAuxClass
objectclass:ibm-idsTransportConditionAuxClass
cn:TrUdpLowPorts
ibm-policyConditionName:TrUdpLowPorts-condition
ibm-idsProtocolRange:17
ibm-idsLocalPortRange:1-1023
ibm-policyKeywords:TR
ibm-policyKeywords:IDS
ibm-policyKeywords:POLICY
description:Reusable IDS TR UDP Low Ports condition
```

Verification

To verify that policies are correctly defined and functioning properly, consider the following points:

- Are the policies active?
- Is the expected traffic mapping to the correct policies?
- Are the IDS Policy functions working correctly?

The following sections provide more details about these considerations.

Are the correct policies active?

Check your LDAP server log or command output for errors encountered when your policies were loaded into LDAP. Some LDAP servers treat consecutive blank lines in an LDIF file as end of file; ensure that all of the policy objects in your LDIF files are acknowledged by LDAP.

Check your Policy Agent log file for errors while processing your policy.

Use the pasearch command to verify that the intended policies are active and have the expected attributes for the target stack.

Is the expected traffic mapping to the correct policies?

Use the netstat -k SUMmary command to ensure that the intended policy has been mapped for each of the Attack types, Scan-Global type and the Scan-Event type for protocol ICMP. Refer to the Refer to *z/OS Communications Server: IP System Administrator's Commands* for more information on the netstat command.

These IDS functions each select the single highest priority policy for their respective types at each policy change.

Use the netstat -k PROTOcol TCP and netstat -k PROTOcol UDP commands to ensure that the intended Scan-Event and TR policies have been mapped to the intended local sockets.

These IDS functions select the highest priority policy for the Protocol, local Port and local IP address when there is relevant activity against the socket.

For TCP this usually entails either a listen or the completion of an inbound connection handshake. For UDP this usually entails either a bind or an inbound datagram. Scan policies are also selected on some inbound error paths.

Are the IDS policy functions working correctly?

IDS policies that include TypeAction:STATISTICS or TypeAction:LOG and Notification:SYSLOG cause the stack to make log record information available to

TRMD. If TRMD is running you may run the IDS report generator TRMDSTAT against the appropriate log files to produce reports on the area of interest.

TRMD

TRMD runs as an authorized program and requires some RACF setup (TRMD must be able to run as a started task and have superuser authority). See the EZARACF member of SEZAINST for sample RACF commands.

The resolver configuration file is used to determine the stack that TRMD will use. Ensure that the RESOLVER_CONFIG environment variable is correctly set before starting TRMD. A separate instance of TRMD must be run for each TCP/IP stack.

The Log records written by TRMD contain 2 timestamps:

- A timestamp generated when the event was detected by the stack. This timestamp is generated by the stack and is always Coordinated Universal Time (UTC).
- A timestamp that is generated when the syslogd record ID is created. This timestamp is dependent on the setting of the TZ environment variable at the time that TRMD is started. If the installation wants this timestamp to be based on UTC, then ensure the TZ environment variable is properly set (for example, export TZ=0) before starting TRMD.

The TCP/IP stack must be running before TRMD can be started.

As described below, TRMD can be started from the z/OS shell or as a started task.

Running TRMD as a started task

A sample procedure is shipped in member EZATRMDP in SEZAINST. Follow the instructions in the sample member to define your environment.

The offset from Coordinated Universal Time (UTC) of the syslog time in the timestamp of TRMD messages is determined by the TZ environment variable. If the timestamp is required in UTC and has not been set by the TZ environment variable, specify the following in the TRMD procedure:

```
//  PARM=('POSIX(ON) ALL31(ON)',
//  'ENVAR("LIBPATH=/usr/lib"',
//  '"TZ=0")/-d 1')
```

To start TRMD as a started task, use the *S TRMD* command from the MVS console or SDSF. In some cases, TRMD issues a fork, and the job name will be the original job name with a number appended. For example, *S TRMD* might result in the TRMD started task running under the job name TRMD1. Use the *D A,TRMD** command to verify the job name that TRMD is running under.

If running as a started task, issue *P jobname* to stop TRMD.

Running TRMD from the z/OS UNIX shell

Only a superuser can run TRMD from the z/OS UNIX shell.

Ensure that the RESOLVER_CONFIG environment variable is correctly set before starting trmd.

The offset from Coordinated Universal Time (UTC) of the syslog time in the timestamp of TRMD messages is determined by the TZ environment variable. If the

timestamp is to appear in Coordinated Universal Time (UTC), change the TZ specification in /etc/profile or export TZ=″0″ before starting TRMD.

After the proper environment is set up, issue the following to start TRMD:

```
trmd
```

Stopping TRMD

To stop TRMD, issue the following kill command :

```
kill -s TERM pid
```

where pid is the TRMD process ID

To obtain the TRMD process ID, issue the following z/OS UNIX command:

```
ps  -A
```

Debug options can also be specified when starting TRMD. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

TRMDSTAT

Trmdstat is a utility program that runs from the z/OS UNIX shell. Trmdstat reads a log file, analyzes the log records generated by TRMD, and provides summary or detailed reports based on the options specified.

The following reports can be requested:

- Overall summary of logged connection events
- IDS summary of logged events
- Reports of logged connection events
- Reports of logged intrusions defined in the ATTACK policy
- Reports of logged intrusions defined in the TCP policy
- Reports of logged intrusions defined in the UDP policy
- Reports of statistics events

Refer to *z/OS Communications Server: IP System Administrator's Commands* for the TRMDSTAT command and samples of the reports generated by TRMDSTAT.

Chapter 14. Network management

This chapter describes how to configure:

- Simple Network Management Protocol Agent (osnmpd)
- z/OS UNIX SNMP command (osnmp)
- NetView® SNMP command
- Subagents
- Trap forwarder daemon

Before you configure, read “Understanding search orders of configuration information” on page 18. It covers important information about data set naming and search sequences. The z/OS UNIX osnmp command is the SNMP command used to access MIB object information from the z/OS shell, as the NetView SNMP command does from NetView.

For an overview of the functional components of SNMP, refer to the SNMP information in *z/OS Communications Server: IP Migration*.

Overview of SNMP

SNMP is a set of protocols that describes management data and the protocols for exchanging that data between heterogeneous systems. The protocols include both the description of the management data, defined in the Management Information Base (MIB), and the operations for exchanging or changing that information. By implementing common protocols, management data can be exchanged between different platforms with relative ease.

Three primary functional entities are defined in SNMP: managers, agents, and subagents. A manager is a management application that typically requests management data. This application could be a simple command-line interface (such as the z/OS UNIX osnmp command and the Netview SNMP command supported in z/OS CS) or a more complex application (such as an MIB browser or a performance monitoring tool). The agent is the server at the host that responds to requests for management data. On z/OS CS, the server is called OSNMPD. The agent is often assisted in its support of management data by one or more subagents that are responsible for providing support for particular sets of management data.

The protocol used to exchange information between SNMP agents and managers is SNMP. The interface used to exchange information between the z/OS CS SNMP agent (and most other agents supported by IBM) and SNMP subagents is the Distributed Protocol Interface (DPI®). The two management commands provided by z/OS CS are the:

- NetView SNMP command
- z/OS UNIX osnmp/snmp command

The NetView SNMP and z/OS UNIX osnmp commands are management applications that can be used to monitor and control network elements. When you use NetView SNMP with TCP/IP, you require the NetView program to provide an end-user interface to the SNMP client. A NetView operator can use the SNMP command to communicate with SNMP agents. NetView acts like an SNMP client. The osnmp command acts like an SNMP client in the z/OS shell.

z/OS CS also provides a special case management application called the Trap Forwarder Daemon to allow multiple managers on z/OS to listen for traps concurrently at the same IP address. The Trap Forwarder Daemon only forwards traps. It does not allow submission of SNMP requests nor does it forward inform-type notifications.

Note: For a list of the MIB objects supported by the SNMP agent and subagents shipped with z/OS CS, refer to *z/OS Communications Server: IP System Administrator's Commands*.

Overview of z/OS CS SNMP version 3

SNMP Version 3 (SNMPv3) is the standards-based solution to previous SNMP security. SNMPv3 is defined in RFCs 2571 through 2575 issued in April 1999 [see Appendix F, "Related protocol specifications (RFCs)" on page 797]. The SNMPv3 architecture is modularized so that portions of it can be enhanced over time without requiring the entire architecture to be replaced. SNMPv3 defines a framework that consists of (among other things):

- A message processing model (SNMPv3)
- A security model (user-based security)
- An access control model (view-based access control)

The framework is structured so that multiple models can be supported concurrently and replaced over time. For example, although there is a new message format for SNMPv3, messages created with the SNMPv1 and SNMPv2c formats can still be supported. Similarly, the user-based security model can be supported concurrently with community-based security models previously used.

Processing an SNMP request

Figure 68 illustrates the interface between TCP/IP and the implementation of SNMP.

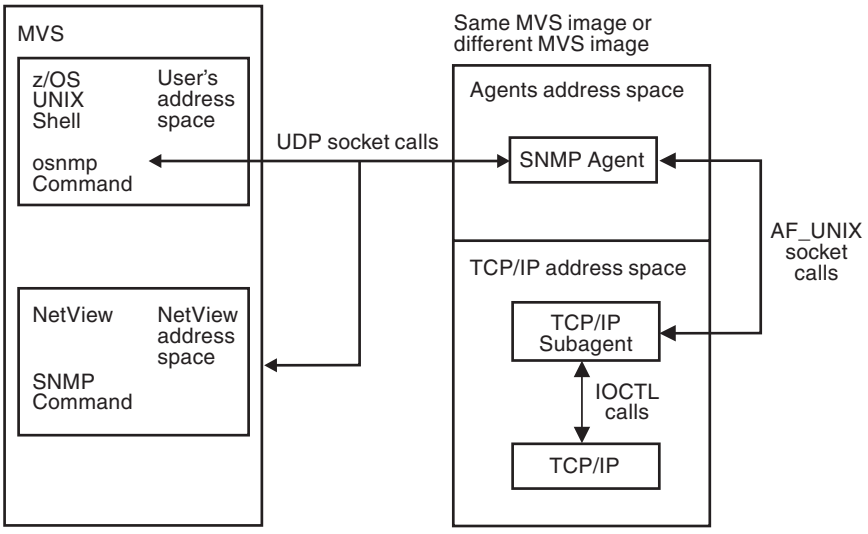


Figure 68. Overview of SNMP support

This list illustrates the sequence of events from the time you issue an SNMP command until you receive the response:

1. The user issues a NetView SNMP (SNMP) or z/OS UNIX SNMP (osnmp) command.

2. The command processor validates and encodes the request in a Protocol Data Unit (PDU), and sends it to the SNMP agent.
3. The SNMP agent validates the request and, if necessary, sends it to an SNMP subagent. Requests for agent-oriented objects are handled by the agent and all others are handled by a subagent. To determine which objects are handled by the agent and which by a subagent, refer to the Management Information Base Appendix in *z/OS Communications Server: IP System Administrator's Commands*.
4. The agent sends the response to the originator of the request. The command processor displays the response.

Note: Although not shown in Figure 68 on page 624, other subagents, such as the OMPROUTE subagent and the SLA subagent shipped as part of z/OS CS, also communicate with the SNMP agent using AF_UNIX socket calls or TCP socket calls from their own address spaces.

The SNMP agent and the SNMP subagents record trace information via the z/OS UNIX syslog daemon using the *daemon* facility. For detailed information regarding syslogd and specifying the daemon facility in the /etc/syslog.conf configuration file, see "Logging of system messages" on page 39.

Deciding on SNMP security needs

The SNMP agent supports SNMPv1, SNMPv2c, and SNMPv3 security. SNMPv1 and SNMPv2c are community-based security, where a community name (or password) is passed with a request. If the community name is recognized as one that can be used by the IP address from which the request originates, the SNMP agent processes the request.

SNMPv3 provides a more powerful and flexible framework for message security and access control. Message security involves providing:

- Data integrity checking, to ensure that the data was not altered in transit
- Data origin verification, to ensure that the request or response originates from the source from which it claims to have come
- Message timeliness checking and, optionally, data confidentiality, to protect against eavesdropping

Access control is the ability to control exactly what data an individual user can read or write.

The SNMPv3 architecture introduces the User-Based Security Model (USM) for message security and the View-Based Access Control Model (VACM) for access control. The architecture supports the concurrent use of different security, access control, and message processing models. For example, community-based security can be used concurrently with USM.

USM uses the concept of a user for which security parameters (levels of security, authentication and privacy protocols, and keys) are configured at both the agent and the manager. Messages sent using USM are better protected than messages sent with community-based security, where passwords are sent in the clear and displayed in traces. With USM, messages exchanged between the manager and the agent have data integrity checking and data origin authentication. Message delays and message replays (beyond what happens normally due to a connectionless transport protocol) are protected against with the use of time indicators and request IDs. Data confidentiality, or encryption, is also available.

The use of VACM involves defining collections of data (called views), groups of users of the data, and access statements that define which views a particular group of users can use for reading, writing, or receipt in a notification.

SNMPv3 also introduces the ability to dynamically configure the SNMP agent using SNMP SET commands against the MIB objects that represent the agent's configuration. This dynamic configuration support enables addition, deletion, and modification of configuration entries either locally or remotely. Remote modification of user keys can be especially useful.

Decide on your security needs—community-based or user-based.

If you are satisfied with the security of your existing configuration, you can continue to use community-based security with no migration. If you would like to take advantage of USM or VACM, you will need to migrate your configuration. Note that USM can be used only when both the SNMP agent and the manager requesting the data support USM, as the z/OS CS SNMP agent and the osnmp command do. VACM can be used even for community-based requests, but doing so requires migration of existing community name and trap destination definitions. Following is a list of the advantages and disadvantages of using each type of security.

Table 22. Security advantages and disadvantages

SNMPv1/SNMPv2c advantages	**SNMPv3 disadvantages**
Traditional standards-based administrative model.	Emerging standards-based administrative model.
Widely implemented on many platforms.	Not yet implemented on many platforms.
Easy to configure.	More robust configuration options.
SNMPv1/SNMPv2c disadvantages	**SNMPv3 advantages**
SNMPv1 and SNMPv2c allow particular IP addresses to access all data or no data.	SNMPv3 allows a particular user to access particular data.
Not very robust (password sent in PDU).	Robust (data integrity and data origin authentication).
Any user that can read data can also change the data (for objects defined as read-write).	The ability to change data can be limited to specific users.
No data confidentiality.	Encryption available.
Configuration changes require restarting of SNMP agent.	Configuration changes for USM and VACM can be made dynamically, either locally or remotely.

For more information about security, see "Creating user keys" on page 633.

Complete the following steps to configure SNMP:

1. Configure the SNMP agent (OSNMPD).
2. Configure the SNMP commands:
   - SNMP for NetView SNMP
   - osnmp for z/OS UNIX SNMP
3. Configure the SNMP subagents.
4. Configure the ATM Open Systems Adapter 2 (ATM OSA) support.
5. Configure the trap forwarder daemon.

Step 1: Configure the SNMP agent (OSNMPD)

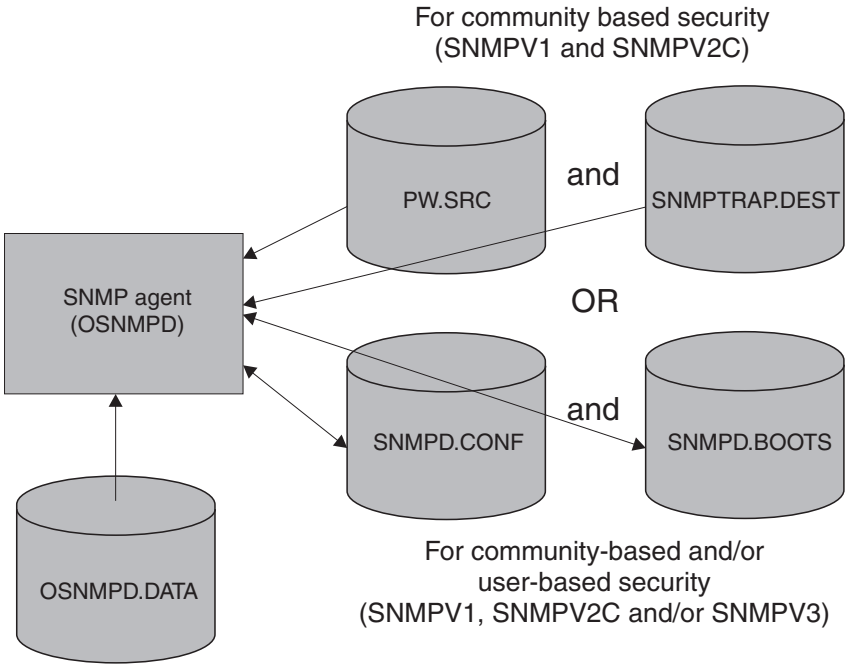


Figure 69. Configuration files for SNMP agent

Configure the SNMP agent based upon your security need. The SNMP agent accepts both SNMPv1 and SNMPv2c requests for community-based security. The SNMP agent can be configured to also use the User-based Security Model and the View-based Access Control Model. To configure the SNMP agent, perform the following tasks:

- "Provide TCP/IP profile statements"
- Depending upon whether you want to use USM and VACM, do one of the following:
  - If you are using community-based security and do not need USM or VACM, see "Provide community-based security and notification destination information" on page 629.
  - If you want the flexibility of using USM or VACM or community-based security, see "Provide community-based and user-based security and notification destination information" on page 631.
- "Provide MIB object configuration information" on page 634
- Refer to *z/OS Communications Server: IP Configuration Reference* for more information about OSNMPD parameters.

Provide TCP/IP profile statements

Update the following configuration statements in *hlq*.PROFILE.TCPIP:

AUTOLOG
PORT

There are two primary TCP/IP ports used by the SNMP agent, one for receiving incoming requests and one for sending traps to managers.

The default port used by the SNMP agent to receive incoming requests is 161. If you want the agent to use port 161 for this purpose and want to insure that no other application uses this port, you must specify the following PORT statement in your profile data set:

```
PORT
  161 UDP OSNMPD              ; SNMP Agent port for SNMP requests
```

If the agent will be started from the z/OS shell, reserve the port instead for z/OS UNIX by typing *OMVS* instead of *OSNMPD*.

If you want to define a port other than 161 for SNMP requests, you must do the following:

1. Start the agent with a `-p` parameter.
2. Configure management applications to use the new port:
   - For the osnmp command, make an entry in the OSNMP.CONF file with the correct port number. For details on creating this entry, see the description for *targetAgent* in the OSNMP.CONF statement in the *z/OS Communications Server: IP Configuration Reference*.
   - Where supported, configure other management applications to use the new port.
3. Configure subagents to use the new port:
   a. Specify the port number to use on the SACONFIG profile statement for the TCP/IP subagent.
   b. Specify the port number to use on the ROUTESA_CONFIG profile statement for the OMPROUTE subagent.
   c. Specify the port number to use on the -p parameter when starting the SLA subagent.
   d. If you are using DPI subagents other than those supplied with z/OS CS, set the SNMP_PORT environment variable to enable user-written subagents to connect to the agent.

The SNMP agent uses port 162, by default, for sending traps to the managers specified in SNMPTRAP.DEST or SNMPD.CONF file. Port 162 should be reserved for the management application primarily responsible for trap processing. If your environment requires multiple management applications at the same IP address to receive traps, consider using the Trap Forwarder Daemon. See “Step 5: Configure the trap forwarder daemon” on page 646 for more details. If the SNMP query engine is typically used for processing traps and other applications, such as osnmp, are only occasionally used, the following port reservations are recommended.

```
PORT
  162 UDP SNMPQE  ; SNMPQuery Engine
```

You must also reserve additional ports for use by the osnmp command by specifying

```
nnnnn UDP OMVS
```

where *nnnnn* is a number in the range 0–65535 and *nnnnn* is used as the -p parameter value on the osnmp trap command.

If you want the SNMPQE and OSNMPD address spaces to be started automatically when the TCPIP address space is started, then include SNMPQE and OSNMPD in the AUTOLOG statement:

```
AUTOLOG
  SNMPQE                    ; SNMP Query Engine
  OSNMPD                    ; SNMP Agent
ENDAUTOLOG
```

Provide community-based security and notification destination information

If you are using only community-based security without the view-based access control model, do the following to configure the security and trap destinations.

Provide community name information

SNMP agents are accessed by remote network management stations. To allow network management stations to send inquiries to the SNMP agent, you may provide PW.SRC information that defines a list of community names and IP addresses that can use these community names. The community name operates as a password when accessing objects on a destination SNMP agent.

The PW.SRC information is optional. If no PW.SRC information is found and no community name is specified for the -c parameter at agent invocation, then the SNMP agent will accept requests with a community name of 'public' from any IP address. If a PW.SRC file exists, but is empty, and if no community name is specified on the `-c` parameter at the agent invocation, then no requests will be accepted by the agent.

Note: Verify that there is no SNMPD.CONF file because this file can only be used with SNMPv3. If an SNMPD.CONF file is found, the PW.SRC file will not be used.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

PW.SRC example: The PW.SRC statements could be specified as follows:

```
passwd1 9.0.0.0      255.0.0.0
passwd2 129.34.81.22 255.255.255.255
```

The PW.SRC statements specify community names and hosts that can use each community name. The format of a statement is:

community_name desired_network snmp_mask

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

The community name of an incoming SNMP request is compared to the known community names. If a match is found, then the IP address of the incoming request is logically ANDed with the *snmp_mask* of the PW.SRC statement. The result of the logical ANDing process is compared with the *desired_network*. If they match, the request is accepted.

In the preceding example, if a request for *community_name* `passwd1` is received from the IP address 9.34.22.122, IP address 9.34.22.122 is ANDed with 255.0.0.0. The result is 9.0.0.0, which equals the specified *desired_network* for `passwd1`, so this request is accepted. In `passwd2`, if the *community_names* match, only requests from host 129.34.81.22 are accepted.

If the *community_name* values do not match, or the IP address ANDed with the *snmp_mask* does not match, an AUTHENTICATION_FAILURE trap is sent if both of the following are true:

- A destination entry exists in SNMPTRAP.DEST.
- Authentication failure traps have been enabled. These traps are enabled by setting MIB object ″snmpEnableAuthenTraps.0″ to 1.

A *desired_network* and *snmp_mask* of all zeros allows anyone with the correct *community_name* to make requests.

Note: By default, the SNMP agent and the osnmp command send packets such that a VIPA address will be used as the originating address in the packet, if SOURCEVIPA is configured. This is a change introduced in V2R10; previously, the SNMP agent and the osnmp command set a socket option to cause the physical interface addresses to be used as the originating addresses on packets they sent. That meant the PW.SRC file had to contain all of the possible physical interface addresses that might be used, rather than a smaller number of VIPA addresses. A customer can override this change in behavior, if desired, by invoking the SNMP agent with the -a option or by using either the -a option or the NOSVIPA option in the osnmp command's OSNMP.CONF configuration file.

Provide trap destination information

Traps are unsolicited messages that are sent by an SNMP agent to an SNMP network management station. An SNMP trap contains information about a significant network event. The management application running at the management station interprets the trap information sent by the SNMP agent.

The following traps are agent-generated:

- Authentication failure
- Cold start

The following traps are generated by the TCP/IP subagent:

- Link down
- Link up
- PVC creation
- PVC deletion

  PVC traps are reported for ATM Permanent Virtual Connections. For further information, see "Step 4: Configure the Open Systems Adapter (OSA) support" on page 642.

The following traps are generated by the SLA subagent:

- Monitored event not achieved for aggregate policy
- Monitored event okay for aggregate policy
- Monitored even not achieved for individual connection policy
- Monitored event okay for individual connection policy
- Policy deleted
- Monitor table entry deleted

Note: The SNMP agent Distributed Protocol Interface allows subagents other than those shipped with z/OS CS (which might be running on another host) to

generate SNMP traps. This can allow for support of other types of traps. For more information about SNMP DPI, see the *z/OS Communications Server: IP Programmer's Reference*.

For additional detail on these traps, refer to the SNMP Trap Types information in *z/OS Communications Server: IP User's Guide and Commands*.

To use traps, you must provide SNMPTRAP.DEST information defining a list of managers to which traps are sent. The SNMPTRAP.DEST information is optional. If no trap destination file is found, then the SNMP agent sends traps to the IP address of the SNMP agent and issues a warning message indicating that defaults are in effect. If a trap destination file exists, but is empty, no traps are sent.

Note: Verify that there is no SNMPD.CONF file. If an SNMPD.CONF file is found, the SNMPTRAP.DEST file will not be used.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

SNMPTRAP.DEST example: The SNMPTRAP.DEST statements could be specified as follows:

```
# SNMP Trap Destination information
124.34.216.1 UDP
MVSSYS2      UDP
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Provide community-based and user-based security and notification destination information

If you want to use user-based security, either concurrently with or instead of community-based security, you must configure security definitions and notification destinations.

SNMPv3 provides the ability to configure the agent dynamically, from either a local or remote host, and to make changes in the configuration while the SNMP agent is running. Doing SNMP agent configuration dynamically requires a good understanding of how the SNMP SET commands can be issued to create new rows or to change or delete existing rows, as well as familiarity with the SNMP engine configuration tables defined in RFCs 2571 through 2576. (For additional detail, see RFCs 2571 through 2576, as well as RFCs 1901 through 1910.)

As an alternative to dynamically configuring the SNMP agent, z/OS CS supports a configuration file to be read at agent initialization called the SNMPD.CONF file. Dynamic configuration changes made with SNMP SET commands to the SNMP agent configuration entries will be written out to the SNMPD.CONF file, so they will continue to be in effect even after the SNMP agent is restarted.

SNMPD.CONF file

The SNMPD.CONF file defines the SNMP agent security and notification destinations. If the SNMPD.CONF file exists, the agent can support SNMPv1, SNMPv2c, and SNMPv3 requests. If no SNMPD.CONF file exists, the agent will support only SNMPv1 and SNMPv2c requests.

Note: If the SNMPD.CONF file is found, the PW.SRC file and the SNMPTRAP.DEST files are not used.

SNMPD.CONF dynamic configuration: If the SNMPD.CONF information is located in an MVS data set rather than an HFS file, special considerations must be made to support dynamic configuration changes to the SNMP agent's configuration. If dynamic configuration changes are made, the file is rewritten to reflect the changes. Therefore, consider the following when allocating the SNMPD.CONF file to an MVS data set:

- The record length (LRECL) should be 512 bytes to accommodate the longest possible entry.
- The use of a member of a partitioned data set is tolerated but not recommended. Because the file might be rewritten often, frequent compression of the partitioned data set may become necessary. In addition, locking on the file is done at the data set level, not at the member level, so other members of the partitioned data set would not be usable while the SNMP agent was running (once a dynamic configuration change had been made).

SNMPD.CONF example: A sample SNMPD.CONF file is shipped as /usr/lpp/tcpip/samples/snmpd.conf.

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

The sample OSNMP.CONF file used by the osnmp command contains entries that match the sample SNMPD.CONF data set. See "Configure the z/OS UNIX SNMP (osnmp) command" on page 640 for additional information on configuring the osnmp command.

Note: By default, the SNMP agent and the osnmp command send packets such that a VIPA address will be used as the originating address in the packet, if SOURCEVIPA is configured. This is a change introduced in V2R10; previously, the SNMP agent and the osnmp command set a socket option to cause the physical interface addresses to be used as the originating addresses on packets they sent. That meant the SNMPD.CONF file had to contain all of the possible physical interface addresses that might be used, rather than a smaller number of VIPA addresses. A customer can override this change in behavior, if desired, by invoking the SNMP agent with the -a option or by using either the -a option or the NOSVIPA option in the osnmp command's OSNMP.CONF configuration file.

SNMPD.BOOTS

The SNMP agent uses the SNMPD.BOOTS configuration file to support SNMPv3 security. This file contains agent information used to authenticate the SNMPv3 requests. The SNMPD.BOOTS keeps the agent identifier and the number of times the agent reboots. If no SNMPD.BOOTS file exists when the agent is started, the agent creates one. You may want to add comments to the beginning of this file. If a file does exist, the agent uses the values specified in the file for setting its engineID and engineBoots values. If the file exists but contains incorrect values for engineID or engineBoots, the agent issues a message and terminates.

Notes:

1. The recommended approach is to allow the SNMP agent to create the file.
2. If the SNMPD.BOOTS file is not provided, the SNMP agent creates the file. If multiple SNMPv3 agents are running on the same MVS image, use the

environment variable to specify different SNMPD.BOOTS files for the different agents. For security reasons, ensure unique engineIDs are used for different SNMP agents.

Creating user keys

Authentication

Authentication is generally required for SNMPv3 requests to be processed (unless the security level requested is 'noAuth'). When authenticating a request, the SNMP agent verifies that the authentication key sent in an SNMPv3 request can be used to create a message digest that matches the message digest created from the authentication key defined for the user.

The osnmp command uses the authentication key found on an entry in the OSNMP.CONF configuration file. It needs to correlate with the authentication key specified on a USM_USER entry for that user in the agent's SNMPD.CONF configuration file.

As an alternative to storing authentication keys in the client configuration file, the osnmp command allows user passwords to be stored. If the osnmp command is configured with a password, the code generates an authentication key (and privacy key if requested) for the user. These keys must, of course, produce the same authentication values as the keys configured for the USM_USER in the agent's SNMPD.CONF file or configured dynamically with SNMP SET commands. However, the use of passwords in the client configuration file is considered less secure than the use of keys in the configuration file.

The authentication key is generated from two pieces of information:

- The specified password.
- The identification of the SNMP agent at which the key will be used. If the agent is an IBM agent and its engineID was generated using the vendor-specific engineID formula, the agent may be identified by IP address or host name. Otherwise, the engineID must be provided as the agent identification.

A key that incorporates the identification of the agent at which it will be used is called a localized key. It can be used only at that agent. A key that does not incorporate the engineID of the agent at which it will be used is called nonlocalized.

Keys stored in the osnmp command's configuration file, OSNMP.CONF, are expected to be nonlocalized keys. Keys stored in the SNMP agent's configuration file, SNMPD.CONF, can be either localized or nonlocalized, though the use of localized keys is considered more secure.

Encryption

Keys used for encryption are generated using the same algorithms as those used for authentication. However, key lengths may differ. For example, an HMAC-SHA authentication key is 20 bytes long, but a localized encryption key used with HMAC-SHA is only 16 bytes long.

z/OS CS provides a facility called *pwtokey* that enables conversion of passwords into localized and nonlocalized authentication and privacy keys. The pwtokey procedure takes as input a password and an identifier of the agent and generates authentication and privacy keys. Since the procedure used by the pwtokey facility is the same algorithm used by the osnmp command, the person configuring the

SNMP agent can generate appropriate authentication and privacy keys to put in the SNMPD.CONF file for a user, given a particular password and the IP address at which the agent will run.

Use the pwtokey command to convert passwords into authentication and privacy keys. Refer to *z/OS Communications Server: IP System Administrator's Commands*.

Provide security product access to agent from subagents

An SNMP subagent can connect to the z/OS Communications Server SNMP agent by using the DPI API (the DPI API is documented in the *z/OS Communications Server: IP Programmer's Reference*) and specifying either a z/OS UNIX or a TCP connection. Subagents using a z/OS UNIX connection are required to have superuser authority. For subagents specifying a TCP connection, you can utilize the installation's SAF compliant security product (such as the z/OS Security Server (RACF)) to control which of the SNMP subagents are permitted to connect to the SNMP agent. One security product resource name can be created per TCP/IP stack per MVS image. The security product resource name is specified in the following format:

```
EZB.SNMPAGENT.sysname.tcpprocname
```

where *sysname* is the name of the MVS system image and *tcpprocname* is the TCP/IP started procedure name.

The profile must be created under the SERVAUTH class. After creating the profiles, use the security product to define the user IDs of those subagents which should be permitted to connect via TCP to the SNMP Agent. Authorization failures are documented by security product failure messages and SNMP agent traces.

Note: If you use this authorization function, only SNMP subagents which are associated with the same TCP/IP stack as the SNMP agent will be permitted to connect to the agent. Local SNMP subagents associated with other TCP/IP stacks, or remote SNMP subagents, will not be permitted to connect. Also, any subagents which connected to the SNMP agent before the agent security product resource name was created will not have been authorized via the security product.

You can use the control statements in the sample JCL job provided in SEZAINST(EZARACF) to define this authorization. For example, if you wanted to permit any SNMP subagents associated with a user ID of USER2 to connect to the SNMP agent you could use the following definitions:

```
RDEFINE SERVAUTH EZB.SNMPAGENT.MVSA.TCP1 UACC(NONE)
PERMIT EZB.SNMPAGENT.MVSA.TCP1  ACCESS(READ) CLASS(SERVAUTH) ID(USER2)
```

Provide MIB object configuration information

An installation can set values for selected MIB objects by providing OSNMPD.DATA information. A sample of OSNMP.DATA is installed as HFS file /usr/lpp/tcpip/samples/osnmpd.data. Refer to *z/OS Communications Server: IP Configuration Reference* for syntax information. If no OSNMPD.DATA file is found, the defaults for these MIB objects are as follows:

Object	Default
dpiPathNameForUnixStream	The default is /tmp/dpi_socket.
sysDescr	If the environment variable HOSTNAME exists, its value is used.

Otherwise, the default value identifies the z/OS system under which the agent is running. The maximum length of this object is 255 octets.

sysContact ″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.

sysLocation ″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.

sysName ″SNMPBASE-Unspecified″. The maximum length of this object is 255 octets.

sysObjectId 1.3.6.1.4.1.2.3.13

Note: sysObjectID is defined as the vendor's authoritative identification of the network management subsystem contained in the entity. That is, it is intended to uniquely identify the SNMP agent. Changing this value is not recommended and will be disabled in a subsequent release.

sysServices A single octet that defaults to 0. See the RFC 1907 description for this object.

snmpEnableAuthenTraps
Default value is 2, which means traps are disabled.

saDefaultTimeout
5 seconds.

saMaxTimeOut
600 seconds.

saAllowDuplicateIDs
Default is 1, which means multiple instances of a subagent are allowed.

Note: Because a subagent identifier cannot be specified for DPI version 1 subagents, a constant identifier is used for all version 1 subagents. Therefore, this object must be set to ″Yes″ (1) to allow multiple DPI version 1 subagents to run concurrently.

For information about where these MIB objects are defined, refer to the *z/OS Communications Server: IP User's Guide and Commands*.

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

Start the SNMP agent (OSNMPD)

The SNMP agent (OSNMPD) runs in a separate address space that executes load module EZASNMPD. OSNMPD can be started with or without parameters. When starting OSNMPD from MVS, add the parameters to the PARMS= keyword on the EXEC statement of the OSNMPD cataloged procedure. When starting OSNMPD from z/OS UNIX, specify the desired parameters on the osnmpd command. Refer to *z/OS Communications Server: IP Configuration Reference* for the command syntax.

Sample JCL procedure for starting OSNMPD from MVS

Update cataloged procedure OSNMPD by copying the sample in *hlq*.SEZAINST(OSNMPDPR) to your system or recognized PROCLIB. Change the data set names as required to suit your local configuration. The OSNMPD address space requires access to the IBM C/370 Library during execution.

Parameters may be passed to the agent on the `PARMS=` keyword on the EXEC statement of the OSNMPD cataloged procedure. Refer to *z/OS Communications Server: IP Configuration Reference* for the command syntax and parameter information. Any agent parameters you wish to specify may be added as shown in the following example:

```
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
// PARM='POSIX(ON) ALL31(ON)/ -c abc -d 255 -p 761'
```

In this example, the agent will use port 761 to accept requests, community name 'abc' will be added to the list of community names supported by the agent, and all agent traces will be activated. For more information on tracing, see the *z/OS Communications Server: IP Diagnosis*.

Starting OSNMPD from z/OS UNIX

To run the SNMP agent in background, you must add an ampersand (&) to the command and the issuer of the command must be in z/OS UNIX superuser mode. For a detailed explanation of the osnmpd parameters, refer to *z/OS Communications Server: IP Configuration Reference*.

Any agent parameters you wish to specify may be added as shown in the following example:

```
osnmpd -c abc -d 255 -p 761
```

In this example, the agent will use port 761 to accept requests, community name 'abc' will be added to the list of community names supported by the agent, and all agent traces will be activated. For more information on tracing, see *z/OS Communications Server: IP Diagnosis*.

Step 2: Configure the SNMP commands

The two SNMP client applications provided with z/OS CS are:

- SNMP command from the NetView environment
- osnmp command in the z/OS shell

The SNMP command in the NetView environment requires the use of the NetView product. It supports SNMP version 1. The osnmp command in the z/OS shell supports SNMP versions 1, 2, and 3. Depending on your requirements, you may decide to configure either or both of these clients, or to use an SNMP client on another platform.

Configure the NetView SNMP (SNMP) command

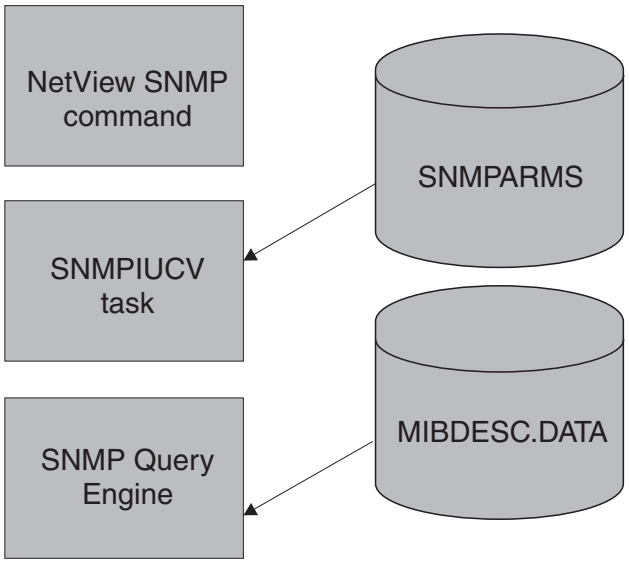


Figure 70. Configuration files for NetView SNMP

The SNMP command in the NetView environment can be used to send SNMP version 1 requests to SNMP agents on either local or remote hosts. The SNMP command requires the command processor itself, the SNMPIUCV task for inter-address space communication, and the SNMP query engine, which creates the packets sent to the SNMP agent. The NetView SNMP command supports only community-based security.

Configure the SNMP query engine

Update the SNMPQE cataloged procedure by copying the sample in *hlq*.SEZAINST(SNMPPROC) to your system or recognized PROCLIB. Specify SNMP parameters and change the data set names as required to suit your local configuration. The SNMPQE address space requires access to the IBM C/370 Library during execution.

The SNMP query engine (SQESERV) needs access to the *hlq*.MIBDESC.DATA data set for the MIB variable descriptions. You can find a sample of this data set in *hlq*.SEZAINST(MIBDESC).

MIBDESC.DATA data set: The MIBDESC.DATA data set defines the short names for MIB variables. Short names are the character representation for the ASN.1 variable names. For example, sysUpTime is the short name for 1.3.6.1.2.1.1.3.0 (the MIB variable that stores the time since the SNMP agent was last restarted). Short names are generally shown as a combination of upper and lowercase characters, though SNMP on z/OS CS ignores these case distinctions. Variable names must always be in ASN.1 language when they are sent to an SNMP agent. You can always use ASN.1 language to specify the variable names in an enterprise-specific tree (assuming that the agent supports them). You can use these short names to specify the MIB variables.

When you issue an SNMP GET, GETNEXT, or SET command, and specify the variable name in ASN.1 notation, the SNMP Query Engine uses that name and sends it in the SNMP packet to the agent. When you issue an SNMP GET, GETNEXT, or SET command, and specify the short name for the variable (for

example, sysDescr), the SNMP Query Engine looks for that name in the MIBDESC.DATA data set and uses the ASN.1 name specified in the data set when it sends the SNMP packet to the agent.

The SNMPQE address space must be able to access the MIBDESC.DATA data set.

You can change the short names in the MIBDESC.DATA data set to the equivalent in your national language. You can also leave the current names and add the equivalent names in your national language. However, the SNMP MIBVNAME function returns only the first entry found in the data set that satisfies the search. In addition, all enterprise-specific variables used by hosts in your network should be added to this data set.

Entries in the data set do not need to be in a specific sequence. Each name starts on a new line. The following shows the line format.

```
short_name asn.1_name type time_to_live
```

Each variable on the line is separated by either one or more spaces or tabs. An asterisk (*) in column 1 indicates that the line is a comment line.

Following is a sample MIBDESC.DATA line with a sysDescr variable translated in Dutch and a few enterprise variables added (in this example, company ABC received 1.3.6.1.4.1.42 as the ASN.1 number for their enterprise):

```
*-------------------------------------------------------------*
* MIB Variable name | ASN.1 notation        | Type     | TTL  *
*-------------------------------------------------------------*
* Following is Dutch name for sysDescr
systeemBeschrijving  1.3.6.1.2.1.1.1.         display    900
sysDescr             1.3.6.1.2.1.1.1.         display    900
  ...
  other entries
  ...
* Following are Enterprise-Specific variables for company ABC
ABCInfoPhone         1.3.6.1.4.1.42.1.1       display    900
ABCInfoAddress       1.3.6.1.4.1.42.1.2       display    900
```

The TTL field contains the number of seconds that a variable lives in the Query Engine's internal cache. If there are multiple requests for the same variable within the TTL period, the variable value is obtained from the cache, and unnecessary network traffic is avoided.

You can define multiple short names or text names for the same variable, as shown with the Dutch translation of the sysDescr variable. In this case, the SNMP Query Engine returns the first value in the table on an SNMP MIBVNAME request. In the previous example, the SNMP Query Engine would return `systeemBeschrijving` and not `sysDescr`. The name returned is in mixed case.

When the SNMP Query Engine receives a short name or text name in a GET, GETNEXT, or SET request, it compares the name against the entries in the MIBDESC.DATA data set. This comparison is not case-sensitive. For example, a request for `SYSDESCR`, `SysDescr`, or `sysDescr` matches the `sysDescr` entry with an ASN.1 notation of `1.3.6.1.2.1.1.1.`.

When the SNMP Query Engine receives an SNMP response, it looks up the variable in the MIBDESC.DATA table Type field for information about translating the value into displayable characters. The information contained in the Type field is case-sensitive and must be specified in lowercase.

Note: If you are using SNMP to receive response or trap PDUs which contain enterprise-specific variables, the variables must be added to the MIBDESC.DATA data set.

Specifying the SNMPQE parameters: The SQESERV module can be configured to start without parameters or you can add any of the following parameters to `PARMS='` in the PROC statement of the SNMPQE cataloged procedure. For example,

```
//SNMPQE   PROC MODULE=SNMPQE,PARMS='-h MVSA'
```

Refer to *z/OS Communications Server: IP Configuration Reference* for the command syntax.

Refer to *z/OS Communications Server: IP Diagnosis* for more information on tracing.

Setting up authorization for SNMPQE: To create RAW sockets necessary for SNMP PING requests, the user ID associated with the SNMPQE started task must have superuser authority (z/OS UNIX UID of 0) or be permitted to BPX.SUPERUSER facility authority.

Configure NetView as an SNMP monitor

To configure the NetView interface as an SNMP monitor, perform each of the following tasks:

- Configure for SNMPIUCV
- Configure for the SNMP command processor
- Configure for the SNMP messages
- Update the SNMP initialization parameters

Configure for SNMPIUCV: SNMPIUCV is the NetView optional task that handles IUCV communication between the NetView program and the SNMP query engine. SNMPIUCV resides in the *hlq*.SEZADSIL data set.

Add the following TASK statement for SNMPIUCV to the DSIDMN member of the data set specified by the DSIPARM DD statement in the NetView start procedure.

```
TASK  MOD=SNMPIUCV,TSKID=SNMPIUCV,PRI=5,INIT=Y
```

This statement causes SNMPIUCV to start automatically when the NetView program is started.

If you specify `INIT=N` instead of `INIT=Y` in the TASK statement for SNMPIUCV, a NetView operator can start the SNMPIUCV task by entering the following:

```
START TASK=SNMPIUCV
```

The SNMPIUCV task tries to connect through IUCV to the SNMP query engine. If this fails, it retries the connect as specified by the SNMPQERT keyword in the SNMPARMS member of the *hlq*.SEZADSIP data set. The default is every 60 seconds.

Configure for the SNMP command processor: SNMP is the command processor that allows NetView operators and CLISTs to issue SNMP commands. SNMP resides in the *hlq*.SEZADSIL data set. This data set should be concatenated to the STEPLIB DD statement in the NetView start procedure.

Add the following statement to the DSICMD member of the data set specified by the DSIPARM DD statement in the NetView start procedure.

```
SNMP  CMDMDL  MOD=SNMP,ECHO=Y,TYPE=R,RES=Y
```

After the SNMPIUCV task is started, you can issue the SNMP command. The SNMP command passes a request to the SNMPIUCV task to forward to SNMPQE. The return code represents a request number that is associated with the request. The responses are returned asynchronously and contain this request number. The operator or CLIST must use the request number to correlate the response to the request.

Configure for the SNMP messages: The NetView SNMP messages reside in the *hlq*.SEZADSIM data set as DSISNM*nn*, where *nn* is the number of the member. The valid message members are DSISNM00 through DSISNM05, DSISNM10, DSISNM12, and DSISNM99. The data set containing these members should be added to the DSIMSG DD statement in the NetView start procedure.

Update the SNMP initialization parameters: SNMPIUCV reads the SNMPARMS member in the *hlq*.SEZADSIP data set at startup. This data set contains the initialization parameters for SNMP. The data set containing SNMPARMS should be added to the DSIPARM DD statement in the NetView startup procedure. Refer to *z/OS Communications Server: IP Configuration Reference* for detailed information for the SNMP parameter data set (SNMPARMS).

Configure the z/OS UNIX SNMP (osnmp) command

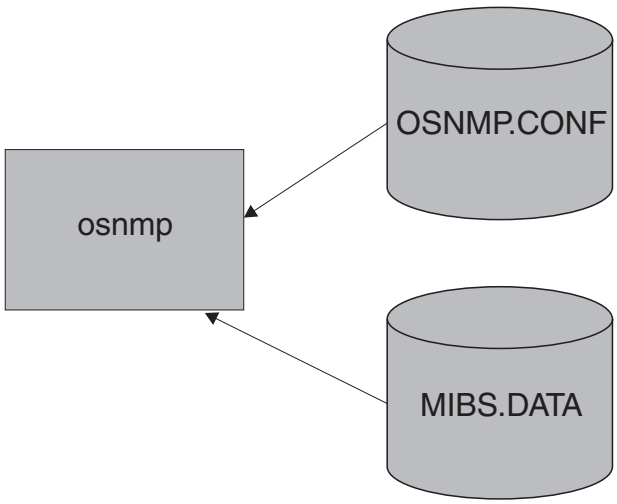


Figure 71. Configuration files for osnmp

The osnmp command is used to send SNMP requests to SNMP agents on local or remote hosts. The requests can be SNMPv1, SNMPv2, or SNMPv3. For SNMPv2 and SNMPv3 requests, the OSNMP.CONF configuration file is required. The *winSNMPname* specified on an OSNMP.CONF statement can be used as the value of the -h parameter on the osnmp command. For a detailed explanation of the parameters you can specify on the osnmp command, see the *z/OS Communications Server: IP System Administrator's Commands*.

To configure the osnmp command, perform the following tasks:

- Provide osnmp configuration information
- Provide user MIB object information

Provide osnmp configuration information

The OSNMP.CONF file is used to define target agents and, for SNMPv3, the security parameters to be used in sending requests to them.

The contents of the file, regardless of location, are the same. Only the first file found is used. A sample of this file is installed as HFS file /usr/lpp/tcpip/samples/snmpv2.conf. This sample should be copied and modified for your installation. Refer to *z/OS Communications Server: IP Configuration Reference* for more information.

Examples:

- Example 1:

  The following entry defines an SNMPv2c node for osnmp:

  ```
  mvs1     9.67.113.79 snmpv2c
  ```

  where *mvs1* is the name used with the -h parameter on the osnmp command and *9.67.113.79* is the IP address of the SNMPv2c agent.

- Example 2:

  The following entry defines an SNMPv3 node:

```
v3mak   127.0.0.1 snmpv3 u1 - - AuthNoPriv  HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987 - -
```

  where *v3mak* is the name used on the -h parameter of the osnmp command. SNMP requests sent using this entry uses USM user name *u1* using HMAC-MD5 authentication but no encryption.

- Example 3:

  The following entry defines an SNMPv3 node. The needed authentication and privacy keys will be generated from the password *u6password*.

```
v3sap   127.0.0.1 snmpv3 u6 u6password - AuthNoPriv   HMAC-SHA -   DES -
```

  The USM user is *u6*. The authentication protocol is HMAC-SHA, and CBC 56-bit DES encryption is used.

Provide user MIB object information

If you want to use the textual names for MIB objects that are not defined in the compiled MIB, then you can define them to the osnmp command using the MIBS.DATA file. All the objects in the list in the Management Information Base (MIB) Objects appendix in the *z/OS Communications Server: IP User's Guide and Commands* are in the compiled MIB. A sample of the MIBS.DATA file is installed as HFS file /usr/lpp/tcpip/samples/mibs.data. Copy this sample and modify it for your installation.

MIBS.DATA statement syntax

The MIBS.DATA statements syntax can be used to specify character (usually called *textual*) names for MIB objects not defined in any compiled MIB supplied with z/OS CS. You can then use these character/textual names as the name of the objects on the osnmp command.

The format of a statement in this file is:

```
character_object_name object_identifier object_type
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Step 3: Configure the SNMP subagents

This section contains information about configuring the TCP/IP subagent.

There are three SNMP subagents shipped with z/OS CS:

- The TCP/IP subagent reports information about the TCP/IP stack.
- The OMPROUTE subagent reports information specific to OSPF. The ROUTESA_CONFIG statement is used in the OMPROUTE configuration file to configure the OMPROUTE subagent. For details on ROUTESA_CONFIG, refer to *z/OS Communications Server: IP Configuration Reference*.
- The SLA subagent reports information about defined service policies and performance statistics related to traffic using those policies. For configuration information about the SLA subagent, see Chapter 12, "Quality of Service (QoS)" on page 565.

There are two statements in the profile data set used to configure the TCP/IP subagent, the SACONFIG and ITRACE statements.

- SACONFIG

  Use the SACONFIG statement to configure the subagent. The SACONFIG parameters determine whether or not the subagent is automatically started at TCP/IP initialization, what port number to use to contact the agent, and other configuration values. For a detailed explanation of this statement, refer to *z/OS Communications Server: IP Configuration Reference*.
- ITRACE

  Use the ITRACE statement to determine what trace information, if any, should be recorded by the subagent. For a detailed explanation of this statement, refer to *z/OS Communications Server: IP Configuration Reference*.

Step 4: Configure the Open Systems Adapter (OSA) support

The TCP/IP subagent can retrieve SNMP management data from the Open Systems Adapter Support Facility (OSA/SF) for several OSA adapters. ATM management data is supported for any OSA-2 ATM or OSA-Express ATM155 adapters. Ethernet management data is supported for any OSA-Express Gigabit Ethernet or OSA-Express QDIO Fast Ethernet adapters. Tables of OSA-Express management data are supported for any OSA-Express Gigabit Ethernet, Fast Ethernet, or ATM155 adapters.

The TCP/IP subagent retrieves the data using the OSA/SF components IOAOSASF (OSA/SF application) and IOASNMP (OSA/SF socket application). For more information on these components, see "OSA/SF prerequisites" on page 644. Specifying the SACONFIG profile statement, with the OSAENABLED and OSASF parameters, in the TCP/IP profile data set causes the TCP/IP subagent to try to connect to the OSA/SF socket application, IOASNMP, using the TCP protocol and the port number specified on the OSASF parameter. If the subagent connects successfully, the following message is issued:

```
EZZ3218I SNMP SUBAGENT: CONNECTED TO OSA/SF
```

Because of timing considerations, the TCP/IP subagent might not be able to connect to IOASNMP at initalization. If this occurs, the subagent will attempt to connect when the first request for OSA MIB data is received. Therefore, the EZZ3218I message might not always be issued during subagent initialization.

When retrieving management data from the OSA adapters, the TCP/IP subagent sends a request to IOASNMP for the data, passing the adapter's portname as an identifier. The portname is obtained from the DEVICE and LINK profile statements used to define the adapter to the TCP/IP stack. The IOASNMP socket application uses APPC to pass the request to the OSA/SF application. The OSA/SF application then retrieves the data from the adapter and returns it to IOASNMP. IOASNMP uses its TCP connection to the TCP/IP subagent to return the data to the subagent. If this configuration is active and either the IOASNMP application or the TCP/IP subagent terminates, the subagent will issue the following message:

```
EZZ3219I SNMP SUBAGENT: DISCONNECTED  FROM OSA/SF
```

To obtain the management data, the adapters must be defined to the TCP/IP stack where the subagent is active, through DEVICE and LINK statements in the TCP/IP profile.

- To retrieve ATM management data, the ATM adapter must be defined as an ATM device/link, even if it is configured for ATM LAN emulation mode and is therefore also defined to some TCP/IP instance as an LCS or MPCIPA device. For OSA-Express ATM155 adapters configured for QDIO LAN Emulation mode, you can use one of the adapter's logical port names on the PORTNAME parameter of the ATM DEVICE statement.
- To retrieve Ethernet management data, the OSA-Express adapter must be defined as an MPCIPA Ethernet link.
- To retrieve OSA-Express management data, the adapters must be defined as the following TCP/IP device/link types:
  - MPCIPA link for Gigabit Ethernet
  - MPCIPA or LCS Ethernet link for Fast Ethernet
  - ATM device/link for ATM155

If the port name is manually configured at the adapter, then management data can be retrieved from the adapter even if it is not active and not in use by any TCP/IP stack or by VTAM. If the port name is dynamically configured (e.g. MPCIPA links), then the adapter has to be active to some TCP/IP stack to retrieve the management data.

Current support consists of:

- Interface table from RFC2233 for ATM, LAN emulation links, AAL5 and ATM layer interfaces.
- ATM Channel table from the IBM MVS Enterprise-Specific MIB for OSA-2 ATM adapters.
- ATM Port and PVC tables from the IBM MVS Enterprise-Specific MIB for OSA-2 ATM and OSA-Express ATM155 adapters .
- ATM LAN Emulation tables from the IBM MVS Enterprise-Specific MIB for OSA-2 ATM and OSA-Express ATM155 adapters.
- atmInterfaceConfTable from RFC1695 for OSA-2 ATM and OSA-Express ATM155 adapters.
- IP over ATM tables from RFC2320 for OSA-2 ATM and OSA-Express ATM155 adapters.
- OSA-Express Channel and Performance tables from the IBM MVS Enterprise-Specific MIB for OSA-Express Gigabit Ethernet, Fast Ethernet, and ATM155 adapters. The performance MIB object values in the ibmMvsOsaExpChannelTable, and all of the MIB object values in the ibmMvsOsaExpPerfTable, are only available starting with zSeries processors where the adapters are, at least, at microcode level 1.31.

- OSA-Express Ethernet Port table from the IBM MVS Enterprise-Specific MIB for OSA-Express Gigabit Ethernet and Fast Ethernet adapters.
- OSA-Express Ethernet SNA table from the IBM MVS Enterprise-Specific MIB for OSA-Express Fast Ethernet adapters.
- dot3StatsTable from EtherLike-MIB in RFC2665 for OSA-Express Gigabit Ethernet and QDIO Fast Ethernet adapters.
- Asynchronous SNMP Trap generation for operational management:
  - ATM Permanent Virtual Circuit (PVC) creation - ibmMvsAtmOsasfAtmPvcCreate Trap (ATM OSA-2 only).
  - ATM Permanent Virtual Circuit (PVC) deletion - ibmMvsAtmOsasfAtmPvcDelete Trap (ATM OSA-2 only).
- Provide method for assigning an IP Address to the ATM Port.

  Use the *osnmp set* command against the ibmMvsAtmOsasfPortIpAddress MIB object to assign the IP Address.

OSA/SF prerequisites

The TCP/IP subagent provided by z/OS CS will connect to OSA/SF to obtain management data. For a subagent to establish a connection to OSA/SF, two OSA/SF components must be started:

- IOAOSASF

  IOAOSASF is a sample JCL procedure that can be used to start the main OSA/SF address space. The sample has a job name of OSASF1.
- IOASNMP

  IOASNMP is a sample JCL procedure that starts the OSA/SF-provided z/OS UNIX socket application that interconnects the TCP/IP subagent with OSASF1.

These sample procedures and all entities that they call are provided with OSA/SF. For a detailed explanation of how to set up OSA/SF on your MVS system, see *OSA Planning*. The primary purpose of OSA/SF is to manage OSA Adapters. It has been extended to support OSA management via SNMP. An instance of IOAOSASF, IOASNMP, the TCP/IP stack, the TCP/IP subagent, and the SNMP agent must be started on every MVS image where OSA management support is needed.

The recommended startup order is:

1. Start IOAOSASF and wait until it completely initializes. IOAOSASF must be started before IOASNMP.
2. Include OSNMPD (the CS SNMP agent) and IOASNMP on the AUTOLOG profile statement for your TCP/IP stack. This ensures that they will be started by autolog processing when TCP/IP is started. For additional profile statement requirements, see “Required TCP/IP profile statements” on page 645. Start the TCP/IP stack.

On an MVS image only a single instance of either IOAOSASF or IOASNMP can (or should) be started. An attempt to start multiple copies of IOAOSASF will be rejected. Starting multiple copies of IOASNMP will yield unpredictable results.

Ensure that OSA/SF is at Version 2 Release 1 level or higher with the OSA/SF APAR OW45237 applied.

Required TCP/IP profile statements

For a detailed description of the statements mentioned here, refer to *z/OS Communications Server: IP Configuration Reference*. The following TCP/IP profile statements must be updated for OSA management support:

- SACONFIG

  On the SACONFIG statement, OSA Management support must be enabled by specifying OSAENABLED. Omission of OSAENABLED when TCP/IP is started will result in no OSA management support. The SACONFIG statement controls the operation of the subagent that runs in a TCP/IP address space as a separate task.

  The OSASF parameter specifies which port IOASNMP should use to Listen for connections from subagents to OSA/SF. For an explanation of the usage of this parameter when starting multiple TCP/IP instances, see "Multiple TCP/IP instances considerations". It is recommended that the OSASF port be reserved by also specifying it on a PORT statement.

  For example:

  ```
  SACONFIG OSAENABLED OSASF 721
  ```

- PORT

  Prereserve the port to be used in communication with OSA/SF.

  For example:

  ```
  PORT
     721  TCP IOASNMP ; OSA/SF TCP/IP Communications
  ```

  In the above example since IOASNMP runs as a z/OS UNIX application the port could have been reserved for z/OS UNIX. Review the /etc/services HFS file to insure that there are no port conflicts.

- DEVICE and LINK

  Provide ATM DEVICE and LINK statements for any OSA ATM adapter for which you want SNMP ATM management data. For example:

  ```
  DEVICE osaName ATM PORTNAME portname
  LINK linkName ATM osaName
  ```

  Provide DEVICE and LINK statements for any OSA-Express Ethernet adapter for which you want SNMP Ethernet management data. For example:

  ```
  DEVICE portname MPCIPA NONROUTER
  LINK linkName IPAQENET portname
  ```

Multiple TCP/IP instances considerations

Subagent connection to OSA/SF

When multiple z/OS CS instances are active in the same MVS image, only one of the TCP/IP instances is connected to IOASNMP. In order for a TCP/IP instance to connect to IOASNMP the OSASF parameter must be specified on the SACONFIG Profile statement.

IOASNMP connects to a TCP/IP instance and acts as a server, receiving connections from those TCP/IP subagents where OSAENABLED was specified on the SACONFIG Profile statement. The result is that all these subagents connect through the same TCP/IP to IOASNMP in order to retrieve OSA information from OSA/SF. For a depiction of this process, see Figure 72 on page 646.

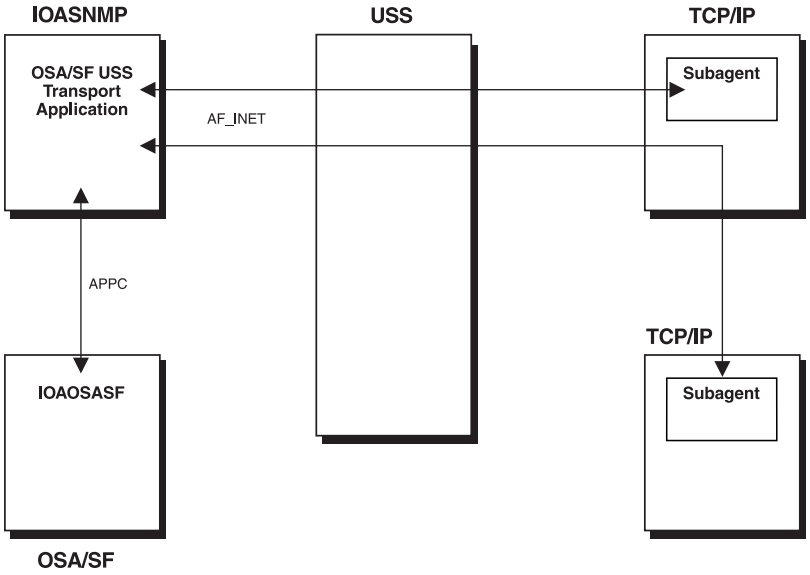


Figure 72. Subagent connection to OSA/SF

If IOASNMP loses its connection to TCP/IP it terminates and needs to be restarted.

If the currently connected TCP/IP terminates, IOASNMP will attempt to connect to another TCP/IP instance for which the OSASF parameter was specified on the SACONFIG Profile statement, using the port number specified for the OSASF parameter. The subagents will also attempt to reconnect to OSA/SF via IOASNMP using this same OSASF port number. For this reason it is recommended that the same OSASF port number be used on the SACONFIG statement of every TCP/IP instance where the OSASF parameter is specified.

Whenever a socket error occurs on the OSA/SF socket, the connected subagents will issue the following message:

```
EZZ3219I SNMP SUBAGENT: DISCONNECTED FROM OSA/SF
```

When the subagent connection is reestablished, the following message is issued:

```
EZZ3218I SNMP SUBAGENT: CONNECTED TO OSA/SF
```

Step 5: Configure the trap forwarder daemon

Most SNMPv1 agents forward traps to port 162. If a manager needs to listen for these traps, it has to bind to port 162 and listen for it. When a manager has already issued a bind it is impossible for another manager to listen for the same traps. The Trap Forwarder daemon eliminates this problem by listening for traps on port 162 and forwarding them to all the configured managers.

You can configure the Trap Forwarder daemon to receive a trap on a specified port and forward it to multiple other ports on the same host and on different hosts. This will allow multiple SNMP managers on z/OS to be able to receive all the traps sent to one port.

To configure the Trap Forwarder daemon, perform the following tasks:

1. Provide PROFILE.TCPIP statements.
2. Provide Trap Forwarder configuration.
3. Start the Trap Forwarder daemon (TRAPFWD).

Provide PROFILE.TCPIP statements

Add or update the following PORT configuration statements in *hlq*.PROFILE.TCPIP.

The default port used by the trap forwarder daemon to receive trap datagrams is UDP port 162. If you want to ensure that no other application uses this port, you must specify the following PORT statement:

```
PORT
  162 UDP TRAPFWD ;  Trap Forwarder daemon
```

If the daemon will be started from the z/OS shell, reserve the port for z/OS UNIX by changing OMVS instead of TRAPFWD. Note that by doing so, the osnmp command could make use of the port if it is started (with the trap option) before TRAPFWD is started.

Provide trap forwarder configuration information

The TRAPFWD.CONF file defines the configuration parameters for trapfwd daemon.

A sample of the TRAPFWD.CONF is shown below:

```
#
# A sample configuration file for trapfwd
#
# Syntax  : ip_address/host_name            port_number            option
#
#
9.67.113.79               1064           ADD_RECVFROM_INFO
myHost                    1066
myHost                    1067           ADD_RECVFROM_INFO
myHost                    1099
9.67.35.37                1064           ADD_RECVFROM_INFO
-                         1065
-                         1068           ADD_RECVFROM_INFO
```

For more information about the statement syntax, refer to *z/OS Communications Server: IP Configuration Reference*.

Starting and stopping the trap forwarder daemon

The Trap Forwarder daemon can be started from the z/OS UNIX shell or from the MVS console.

Starting the trap forwarder daemon from z/OS UNIX

This example starts the Trap Forwarder daemon on the standard port (port 162).

```
# trapfwd
```

This example starts the Trap Forwarder daemon on a particular port (port 5062).

```
# trapfwd -p 5062
```

Starting the trap forwarder daemon from an MVS console: Update cataloged procedure TRAPFWD by copying the sample in hlq.SEZAINST(EZASNTRA) to your system. Refer to *z/OS Communications Server: IP Configuration Reference* the for more information about this cataloged procedure.

Stopping the trap forwarder daemon:

From MVS:

```
P TRAPFWD
```

If TRAPFWD was started from a cataloged procedure, procname is the member name of that procedure. If TRAPFWD was started from the z/OS shell, procname is useridX, where X is the sequence number set by the system. To determine the sequence number issue d omvs u=userid from the console. This will show the programs running under the user ID.

From UNIX:

```
kill < pid >
```

Issue the kill command to the process ID (PID) associated with TRAPFWD. To find the PID issue the *ps -ef* command from the z/OS shell.

Tracing: The modify command can be used to display the existing tracing level and also to change the tracing level.

The following example sets the trace level of the Trap Forwarder Daemon to 1.

```
MODIFY TRAPFWD,TRACE,LEVEL=1
```

The following example displays the level of tracing set for the Trap Forwarder Daemon.

```
MODIFY TRAPFWD,TRACE,QUERY
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Dynamically refreshing configuration: The modify command can be used to dynamically refresh the configuration information. When this is done, the old configuration information is discarded completely. The configuration file is read again and the daemon is initialized.

The following example refreshes the configuration by reading the configuration file.

```
MODIFY TRAPFWD,REFRESH
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about syntax.

Chapter 15. Remote Print Server (LPD)

Read “Understanding search orders of configuration information” on page 18. It covers important information about data set naming and search sequences.

The Remote Print (LPD) server supports the Line Print Daemon and allows you to print on JES controlled printers from any host in your TCP/IP network that implements the Line Print client functions. These client functions are invoked with the LPR command. LPR is available as a TSO command, and the LPD server is implemented as a started z/OS task.

Refer to the *z/OS Communications Server: IP System Administrator's Commands* for information on starting and stopping the TCP/IP print server (LPD). When LPD is stopped by the MVS operator with the *P procname* command, LPD will terminate any TCP/IP connections currently transferring data. Before ending, LPD will finish spooling to JES any print jobs that it has received and is currently spooling. JES will handle these jobs after LPD ends.

This chapter describes how to configure the LPD server. To operate the LPD server after it is running, refer to *z/OS Communications Server: IP Configuration Reference*.

The (SMTP or LPD) server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E will be issued and the server will terminate.

Configuring the Remote Print Server

Steps to configure the Remote Print Server:

1. Specify AUTOLOG and PORT statements in PROFILE.TCPIP.
2. Update the LPD server cataloged procedure.
3. Update the LPD server configuration data set.
4. Create a banner page (optional).

For information about operating and controlling the LPD server, refer to *z/OS Communications Server: IP Configuration Reference*.

Step 1: Configuring PROFILE.TCPIP for LPD

If you want the LPD server started automatically when the TCPIP address space is started, include the name of the member containing the LPD server cataloged procedure in the AUTOLOG list in *hlq*.PROFILE.TCPIP. For example:

```
AUTOLOG
  LPSERVE
ENDAUTOLOG
```

To ensure that port TCP 515 is reserved for the LPD server, also add the name of the member containing the LPD cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP. For example:

```
PORT
  515 TCP LPSERVE
```

See the *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Updating the LPD server cataloged procedure

Update the LPD server cataloged procedure to suit your local configuration by copying the sample to your system or recognized PROCLIB from *hlq*.SEZAINST(LPSPROC) and modifying the sample. Specify LPD parameters, and change the data set names as required. See "Specifying LPD server parameters" for more information on the LPD server parameters.

Specifying LPD server parameters

The system parameters required by the LPD server are passed by the PARM option on the EXEC statement of the LPD cataloged procedure. Update the following parameters as required.

LPDDATA='*data_set_name***'**
Specifies the fully qualified name of the data set containing the LPD configuration statements. This data set can be sequential or a member of a PDS.

LPDPRFX='PREFIX *your_prefix***'**
Specifies the high-level qualifier to be used for temporary data sets created by the LPD server. Include both the PREFIX keyword and your qualifier in the quoted string. The qualifier may be up to 26 characters. If it is blank, it defaults to the procedure name. The LPD task requires the authority to create and modify data sets with this prefix.

DIAG='*options***'**
Specifies any of the following diagnostic options in a quoted string of keywords separated by blanks. For example, `DIAG='VERSION TRACE'`

VERSION
Displays the version number.

TYPE Activates high-level trace facility in the LPD server. Significant events, such as the receipt of a job for printing, are recorded in the //SYSOUT DD data set specified in your LPD server cataloged procedure.

TRACE
Causes a detailed trace of activities within the LPD server to record in the //SYSOUT DD data set specified in your LPD server cataloged procedure. The detailed tracing can also be activated with the DEBUG statement in the LPD server configuration data set and with the TRACE command of the SMSG interface.

Note: The JCL PARM= statement has a limit of 100 characters.

Configuring LPDDATA

Use a DD card that requires the LPD configuration file to be in a data set.

```
//SYSLPDD  DD DISP=SHR,DSN=TCPIP.V2R7.SEZAINST(LPDDATA)
```

Using this example, the DD card must be specified as SYSLPDD, but the data set name can be any valid data set name with a member specified up to 44 characters.

In order to use the DD card method, you *must* comment out the LPDDATA= and remove ″&LPDDATA″; from the PARM= statement.

Note: The search order for the configuration file is:
1. LPDDATA= on the PARM= statement
2. //SYSLPDD DD statement
3. *hlq*.LPD.CONFIG

If both the LPDDATA= statement on the PARM= statement and the //SYSLPDD DD statement are specified, the data set name specified on LPDDATA= is used.

The LPD server does not limit the number of print jobs it handles per connection. This can cause a memory abend to occur if too many print jobs are sent in one connection. Certain LPR clients, such as SUN UNIX, are set up to send multiple jobs in one connection. It is recommended that the LPD start procedure be started with a region size of 6M and the LPR client send no more than 50 print jobs in one connection.

Step 3: Updating the LPD server configuration data set

The LPD configuration data set defines the local, NJE, and remote services (printers and punches) used by the LPD server. To update the LPD server configuration data set, copy the sample provided in *hlq*.SEZAINST(LPDDATA) and modify it to suit your installation. Refer to *z/OS Communications Server: IP Configuration Reference* for more details.

- To define a printer or punch:
  - Include a SERVICE statement with the appropriate parameters for each printer or punch your are defining.
  - Specify the type of service with the LOCAL, NJE, or REMOTE parameter in the SERVICE statement.
  - For local or NJE services, include any of the optional parameters to further define the service: EXIT, FAILEDJOB, FILTERS, LINESIZE, PAGESIZE, RACF, and SMTP.
  - For remote services, specify the destination printer and host. Any additional specifications are defined on the remote system and are not necessary in the SERVICE statement.
- To turn on LPD server tracing, include the optional DEBUG statement.
- To authorize users for the SMSG interface, include the optional OBEY statement.
- Printer names cannot contain an at sign (@).

Step 4: Creating a banner page (optional)

LPBANNER is the name of the default program that is provided in executable form in the SEZATCP data set. This program is specified on the EXIT parameter in the SERVICE statement. LPBANNER prints a separator page that contains, in large letters, a banner stating “LPD BANNER”, the user ID, the job name, and the job class. Field headings of HOST, USER, JOB, and CLASS appear in smaller letters.

The sample exit LPBANNER uses machine carriage control and is designed to be used with the SERVICE PRINTER LOCAL or SERVICE PRINTER NJE statements. Banner pages are usually not used with the SERVICE PUNCH or SERVICE RECFMU statements; however, if you want to have banner pages (headers) for the SERVICE PUNCH or SERVICE RECFMU devices, a user created banner exit is required.

You can either use the executable form or copy and modify the sample source provided in *hlq*.SEZAINST(EZAAE04S) and *hlq*.SEZAINST(EZAAE04T) to create a banner. If you are changing the source to create your own banner, assemble and link-edit these data sets as reentrant. You can modify and use the following JCL to do this. Changing the ENTRY and NAME to something other than LPBANNER will avoid possible maintenance problems in the future.

```
//ASMLNK JOB MSGLEVEL=(1,1),MSGCLASS=A,CLASS=A,REGION=1024K
//ASM1   EXEC PGM=ASMA90,PARM='OBJECT,XREF(FULL)'
//STEPLIB DD  DISP=SHR,DSN=HLA.OSV1R4.SASMMOD1
//*  ASSEMBLER H
//SYSLIB   DD DSN=tcpip..V3R4.SEZACMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DSN=&&SYSUT1
//SYSPUNCH DD DUMMY,DCB=BLKSIZE=80
//SYSPRINT DD SYSOUT=A
//SYSLIN   DD DSN=&&OBJECT(EZAAE04S),DISP=(,PASS),UNIT=SYSDA,
//            SPACE=(CYL,(5,5,1)),DCB=BLKSIZE=400
//SYSIN    DD DSN=tcpip.V3R4.SEZAINST(EZAAE04S),DISP=SHR
/*
//ASM2   EXEC PGM=ASMA90,PARM='OBJECT,NODECK,XREF'
//STEPLIB  DD  DISP=SHR,DSN=HLA.OSV1R4.SASMMOD1
//*  ASSEMBLER H
//SYSLIB   DD DSN=tcpip.V3R4.SEZACMAC,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DSN=&&SYSUT1
//SYSPUNCH DD DUMMY,DCB=BLKSIZE=80
//SYSPRINT DD SYSOUT=A
//SYSLIN   DD DSN=&&OBJECT(EZAAE04T),DISP=(OLD,PASS)
//SYSIN    DD DSN=tcpip.V3R4.SEZAINST(EZAAE04T),DISP=SHR
/*
//LNK    EXEC PGM=IEWL,PARM='LIST,NCAL,RENT,LET'
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD DSN=tcpip.V3R4.SEZATCP,DISP=SHR
//AEZAMODS DD DSN=tcpip.V3R4.AEZAMODS,DISP=SHR
//OBJECT   DD DSN=&&OBJECT,DISP=(OLD,DELETE)
//SYSLIN   DD *
  ORDER   EZBOECPR
  INCLUDE AEZAMODS(EZBOECPR)
  INCLUDE OBJECT(EZAAE04S)
  INCLUDE OBJECT(EZAAE04T)
  MODE    AMODE(24),RMODE(24)
  ENTRY   LPBANNER
  NAME    LPBANNER(R)
/*
```

Chapter 16. Remote procedure calls

This chapter contains information to help you configure:

- PORTMAP
- UNIX PORTMAP
- NCS
- NDB

Read “Understanding search orders of configuration information” on page 18. It covers important information about data set naming and search sequences.

Configuring the PORTMAP address space

This section describes how to configure the PORTMAP address spaces, which runs the Portmapper function.

Portmapper is the software that supplies client programs with the port numbers of server programs. Clients contact server programs by sending messages to port numbers where receiving processes receive the message. Because you make requests to the port number of a server rather than directly to a server program, client programs need a way to find the port number of the server programs they wish to call. Portmapper standardizes the way clients locate the port number of the server programs supported on a network.

Steps to configure PORTMAP:

1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the PORTMAP cataloged procedure.
3. Define the data set for well-known procedure names.

Step 1: Configuring PROFILE.TCPIP for PORTMAP

If you want the PORTMAP server to start automatically when the TCPIP address space is started, you should include PORTMAP in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  PORTMAP
ENDAUTOLOG
```

Note: If your system is using the Network File System (NFS) server, you ***must*** start the PORTMAP address space. See *z/OS Network File System Customization and Operation* for more information.

To ensure that port UDP 111 and TCP 111 are reserved for the PORTMAP server, also add the name of the member containing the PORTMAP cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  111 UDP PORTMAP
  111 TCP PORTMAP
```

See the *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Updating the PORTMAP cataloged procedure

Update the PORTMAP cataloged procedure to suit your local conditions by copying the sample provided in *hlq*.SEZAINST(PORTPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Change the data set names as required. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details.

Step 3: Defining the data set for well-known procedure names

z/OS CS includes a data set that contains a list of commonly used procedure names. This data set is used by the RPCINFO command to resolve remote program numbers into their equivalent program names.

To create the data set, copy the ETCRPC member of *hlq*.SEZAINST to the default data set called *hlq*.ETC.RPC. If a user has a *user_id*.ETC.RPC data set defined, it takes precedence over the preceding data set.

Normally, you would not change this data set except to add a new application to the list. To add a new application, add a line that contains the following items:

- The *program_name* of the new application or procedure
- The *program_number* of the new application or procedure
- Any comments regarding the description of the program

The items are variable in format, each separated by a blank.

The *hlq*.SEZAINST(ETCRPC) data set contains the well-known procedure names. Following is the ETCRPC sample.

```
#
#   rpc   1.2   86/10/07
#
#   COPYRIGHT = NONE.
#
portmapper    100000    portmap sunrpc
rstatd        100001    rstat rup perfmeter
rusersd       100002    rusers
nfs           100003    nfsprog
ypserv        100004    ypprog
mountd        100005    mount showmount
ypbind        100007
walld         100008    rwall shutdown
yppasswdd     100009    yppasswd
etherstatd    100010    etherstat
rquotad       100011    rquotaprog quota rquota
sprayd        100012    spray
3270_mapper   100013
rje_mapper    100014
selection_svc 100015    selnsvc
database_svc  100016
rexd          100017    rex
alis          100018
sched         100019
llockmgr      100020
nlockmgr      100021
x25.inr       100022
statmon       100023
status        100024
#
# NDB Program numbers added for more useful rpcinfo output
# Values are in decimal. Corresponding hexadecimal values
# for the NDB port manager is X'20000020' and for the NDB
# servers are from X'20000021' to X'20000084'.
#
```

```
ndbportmgr    536870944
ndbserver1    536870945
ndbserver2    536870946
ndbserver3    536870947
ndbserver4    536870948
ndbserver5    536870949
ndbserver6    536870950
ndbserver7    536870951
ndbserver8    536870952
ndbserver9    536870953
ndbserver10   536870954
ndbserver11   536870955
ndbserver12   536870956
ndbserver13   536870957
ndbserver14   536870958
ndbserver15   536870959
ndbserver16   536870960
ndbserver17   536870961
ndbserver18   536870962
ndbserver19   536870963
ndbserver20   536870964
ndbserver21   536870965
ndbserver22   536870966
ndbserver23   536870967
ndbserver24   536870968
ndbserver25   536870969
ndbserver26   536870970
ndbserver27   536870971
ndbserver28   536870972
ndbserver29   536870973
ndbserver30   536870974
ndbserver31   536870975
ndbserver32   536870976
ndbserver33   536870977
ndbserver34   536870978
ndbserver35   536870979
ndbserver36   536870980
ndbserver37   536870981
ndbserver38   536870982
ndbserver39   536870983
ndbserver40   536870984
ndbserver41   536870985
ndbserver42   536870986
ndbserver43   536870987
ndbserver44   536870988
ndbserver45   536870989
ndbserver46   536870990
ndbserver47   536870991
ndbserver48   536870992
ndbserver49   536870993
ndbserver50   536870994
ndbserver51   536870995
ndbserver52   536870996
ndbserver53   536870997
ndbserver54   536870998
ndbserver55   536870999
ndbserver56   536871000
ndbserver57   536871001
ndbserver58   536871002
ndbserver59   536871003
ndbserver60   536871004
ndbserver61   536871005
ndbserver62   536871006
ndbserver63   536871007
ndbserver64   536871008
ndbserver65   536871009
ndbserver66   536871010
```

```
ndbserver67   536871011
ndbserver68   536871012
ndbserver69   536871013
ndbserver70   536871014
ndbserver71   536871015
ndbserver72   536871016
ndbserver73   536871017
ndbserver74   536871018
ndbserver75   536871019
ndbserver76   536871020
ndbserver77   536871021
ndbserver78   536871022
ndbserver79   536871023
ndbserver80   536871024
ndbserver81   536871025
ndbserver82   536871026
ndbserver83   536871027
ndbserver84   536871028
ndbserver85   536871029
ndbserver86   536871030
ndbserver87   536871031
ndbserver88   536871032
ndbserver89   536871033
ndbserver90   536871034
ndbserver91   536871035
ndbserver92   536871036
ndbserver93   536871037
ndbserver94   536871038
ndbserver95   536871039
ndbserver96   536871040
ndbserver97   536871041
ndbserver98   536871042
ndbserver99   536871043
ndbserver100  536871044
```

Starting the PORTMAP address space

If you did not include PORTMAP in the AUTOLOG statement, you can start PORTMAP with the START command. For example:

```
START PORTMAP
```

PORTMAP must be started before NDB can run.

Configuring the z/OS UNIX PORTMAP address space

This section describes how to configure the z/OS UNIX PORTMAP address space, which runs the Portmapper function.

Steps to configure z/OS UNIX PORTMAP:

1. Specify PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the PORTMAP cataloged procedure.

Step 1: Configuring PROFILE.TCPIP for UNIX PORTMAP

If you want the PORTMAP server to start automatically when the TCPIP address space is started, you should include PORTMAP in the AUTOLOG statement in the hlq.PROFILE.TCPIP data set.

```
AUTOLOG
  PORTMAP JOBNAME PORTMAP1
ENDAUTOLOG
```

To ensure that port UDP 111 and TCP 111 are reserved for the z/OS UNIX PORTMAP server, add the z/OS UNIX PORTMAP server jobname to the PORT statement in *hlq*.PROFILE.TCPIP. If you use the sample cataloged procedure, PORTMAP, to start the z/OS UNIX PORTMAP server, the jobname is PORTMAP1:

```
PORT
  111 UDP PORTMAP1     ; Portmapper Server
  111 TCP PORTMAP1     ; Portmapper Server
```

See the *z/OS Communications Server: IP Configuration Reference* for more information about the PORT statement.

Step 2: Updating the PORTMAP cataloged procedure

Update the PORTMAP cataloged procedure to suit your local conditions by copying the sample provided in *hlq*.SEZAINST(OPORTPRC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Change the data set names as required. Refer to the *z/OS Communications Server: IP Configuration Reference* for more details.

Starting the PORTMAP address space

There are two ways to start the portmapper as a z/OS UNIX socket application:

- From the z/OS shell
- As a started task

To start the portmapper from the z/OS shell, the user ID must be an authorized superuser. The authorized superuser ID can issue *oportmap &* to start the portmapper. For the authorization procedure, see *z/OS UNIX System Services Planning*.

You can also start PORTMAP as a started task with the START command as follows:

```
START PORTMAP
```

Note: If your system is using the Network File System (NFS) server, see *z/OS Network File System Customization and Operation* for more information.

Configuring the NCS interface

This section describes how to configure the Network Computing System (NCS).

NCS is the Remote Procedure Call (RPC) implementation of Apollo's Network Computing Architecture (NCA**).

NCS includes:
1. RPC run-time library
2. Location broker
3. Network Interface Definition Language (NIDL) compiler

The RPC run–time library and the location broker provide runtime support. Together these two elements make up the Network Computing Kernel (NCK) which includes all the software necessary to run a distributed application. The NIDL compiler is a tool for developing NCS-based applications.

In order to execute NCS applications in an MVS environment, you must start a local location broker (LLBD). One of the hosts in your TCP/IP network must also start the

global location broker (GLBD). Both the LLBD and the NRGLBD maintain information about active NCS server applications.

Understanding the LLBD server

The LLBD manages the LLB database which stores information for the NCS-based servers running on this host.

Your host must run LLBD if you want to support the location broker forwarding function or allow remote access to the LLB database. The LLBD function must be started on the host before any other NCS-based servers are started and before the NRGLBD is started.

The LLB database is stored in the data set ADM@SRV.LLB@LL.DATABASE, which is not created until an entry is registered with the LLBD. This data set can be administered using the lb@admin tool.

Understanding the NRGLBD server

The NRGLBD manages the NCS global location broker (GLB) database. The GLB helps clients locate servers on a network or internet.

There are two versions of the GLB daemon: replicated GLBD and NRGLBD. The replicated GLBD is only available on Apollo, SunOS, and Ultrix systems. For other systems, such as IBM, only the NRGLBD is available. The advantage of replicated GLBD over NRGLBD is that the GLBD can be run at the same time on several network hosts, providing greater availability in the event that one of the hosts running GLBD fails or if there is a partial network failure.

You cannot use the NRGLBD on your system if:
- The replicated version of GLBD can run on some other host in your network
- Another host in your network is already running NRGLBD

The GLB database is stored in a data set ADM@SRV.LLB@LG.DATABASE. This data set is not created until an entry is registered with the NRGLBD.

The record structure for the LLBD and the NRGLBD databases is identical.

Steps to configure NCS:
1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the NRGLBD cataloged procedure in *hlq*.SEZAINST(NRGLBD).
3. Update the LLBD cataloged procedure in *hlq*.SEZAINST(LLBD).

For more information about NCS, see *Network Computing System Reference Manual*.

Step 1: Configuring PROFILE.TCPIP for NCS

If you want LLBD and NRGLBD to start automatically when the TCPIP address space is started, then you should include the names of the members containing the LLBD and NRGLBD cataloged procedures in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

The LLBD must be running before you start NRGLBD. Therefore, you must put LLBD before NRGLBD in the AUTOLOG statement.

```
AUTOLOG
  LLBD
  NRGLBD
ENDAUTOLOG
```

To ensure that port UDP 135 is reserved for the LLBD server, add the name of the member containing the LLBD cataloged procedure to the PORT statement in the *hlq*.PROFILE.TCPIP data set.

```
PORT
  135 UDP LLBD
```

Note: z/OS UNIX DCE also uses port 135 and therefore cannot be used concurrently with NCS.

See the *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Updating the NRGLBD cataloged procedure

Update the NRGLBD cataloged procedure by copying the sample provided in *hlq*.SEZAINST(NRGLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions. Refer to the NCS chapter in *z/OS Communications Server: IP Configuration Reference* for more details.

Step 3: Updating the LLBD cataloged procedure

Update the LLBD cataloged procedure by copying the sample provided in *hlq*.SEZAINST(LLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions. Refer to the NCS chapter in *z/OS Communications Server: IP Configuration Reference* for more information.

Configuring the Network Database (NDB) System

This section describes how to configure the Network Database (NDB) System. NDB provides access to mainframe relational database systems using TCP/IP and the remote procedure call (RPC) protocol. This allows interoperability among a variety of workstation and mainframe users with DB2 and MVS. End users in a VM, MVS, DOS, OS/2, RISC System/6000 AIX, or Sun UNIX Microsystems environment can issue SQL statements interactively or invoke NDB services from within a C application program. NDB services can then be used to pass SQL statements to DB2 and to handle responses from DB2.

The Network Database (NDB) System consists of:

- The Network Database (NDB) Port Manager
- The NDB Port Client
- 1–100 NDB Servers
- NDB Clients

Note: The NDB System requires the Portmapper to run. See “Configuring the PORTMAP address space” on page 653 for further information.

Steps to Configure the NDB System

1. Update the NDB setup sample job (NDBSETUP).
2. Run the NDB setup sample job.
3. Update and install the DB2 sample connection exit routine (DSN3SATH), if necessary.

4. Update the PORTS cataloged procedure (PORTSPRC).
5. Update the PORTC cataloged procedure (PORTCPRC).
6. Create the NDB clients.

Note: Repeat steps 1 and 2, and step 3 if necessary, to configure each DB2 subsystem that is to be accessed by NDB servers.

Step 1: Updating the NDB setup sample job

Update the NDBSETUP sample job by copying the sample in *hlq*.SEZAINST(NDBSETUP) to a partitioned data set (PDS) or sequential data set and modifying it to suit your local installation. Change the DD statements as required. A copy of NDBSETUP can also be found *z/OS Communications Server: IP Configuration Reference*.

To restrict the use of NDB to selected users, modify the GRANT statement in NDBSETUP job replacing PUBLIC with a list of the specified TSO user IDs to whom you want to grant access. See the *DB2 SQL Reference* for correct statement syntax.

THE NDBSETUP job binds the DBRM DBUTIL2 into the DB2 subsystem specified. In previous releases, the DBRM was bound using the plan name DBUTIL2. For V3R2, the DBRM is bound using the plan name EZAND320. By using different plan names for each release, you can have different levels of TCP/IP simultaneously accessing the same DB2 subsystems.

Note: NDBSETUP binds the DBRM DBUTIL2 with the isolation level of cursor stability: ISOLATION(CS) on the BIND command. If a higher degree of unit of work integrity is needed than is provided by an isolation level of cursor stability, this can be changed to an isolation level of repeatable read: ISOLATION(RR) on the BIND command. Refer to the *DB2 Application Programming and SQL Guide* for information regarding isolation level settings and their effects. Check with your database administrator regarding isolation level policies at your site.

Step 2: Running the NDB setup job

Run NDBSETUP to bind the DBRM DBUTIL2 to the DB2 subsystem specified and grant the execute (run) privileges to public. Verify that it ran successfully before proceeding with this configuration.

Notes:

1. The DB2 subsystem must be Version 2 Release 3 or higher. This requirement applies to all TCP/IP servers that use DB2.
2. The DB2 subsystem must be up and running before you submit the NDBSETUP job.
3. The NDBSETUP job must be executed from a user ID with the authority to bind plans for the specified DB2 subsystem.

Step 3: Updating and installing the DB2 sample connection exit routine

Note: If you will be running only one NDB server in a single address space, you may skip this step.

Because the address space running a PORTC cataloged procedure is capable of running between 1 to 20 NDB servers, changes must be made in the DB2 sample

connection exit routine, DSN3SATH ASSEMBLE. These changes tell DB2 where to look for the host userid to be used for this session (the host userid was supplied by the user at NDB client invocation).

After making the necessary changes in DSN3SATH ASSEMBLE, this routine can be assembled and link edited using the DB2 supplied job DSNTIJEX in the DSNSAMP library. See DSN3SATH ASSEMBLE modifications for NDB for the updated assembler code. For more information, see the *DB2 Administration Guide*. DSNTIJEX will place a copy of the executable in the DSNEXIT library. There is one DSNEXIT library for each DB2 subsystem and it must be updated with the recommended modifications for each DB2 subsystem that will be accessed via NDB. Run DSNTIJEX once (updating the hlq for the DSNEXIT library) for each DB2 subsystem that will be accessed via NDB.

DSN3SATH ASSEMBLE modifications for NDB

```
⋮
**********SECTION 1:  DETERMINE THE PRIMARY AUTHORIZATION ID  *********
*                                                                     *
*   IF THE INPUT AUTHID IS NULL OR BLANKS, CHANGE IT TO THE AUTHID    *
*   IN EITHER THE JCT OR THE FIELD POINTED TO BY ASCBJBNS.            *
*                                                                     *
*   THE CODE IN THIS SECTION IS AN ASSEMBLER LANGUAGE VERSION OF      *
*   THE DEFAULT IDENTIFY AUTHORIZATION EXIT.  IT IS EXECUTED ONLY     *
*   IF THE FIELD ASXBUSER IS NULL UPON RETURN FROM THE RACROUTE       *
*   SERVICE.  FOR EXAMPLE, IT DETERMINES THE PRIMARY AUTH ID FOR      *
*   ENVIRONMENTS WITH NO SECURITY SYSTEM INSTALLED AND ACTIVE.        *
*                                                                     *
***********************************************************************
         SPACE
* MOVED CHECK ON AIDLPRIM THAT WAS HERE TO PAST CHECK FOR TSO FOREGRND
         L     R7,ASCBCSCB         GET CSCB ADDRESS
         CLI   CHTRKID-CHAIN(R7),CHTSID IS IT TSO FOREGROUND ADDR SPACE
         BNE   SATH010             BRANCH IF NOT
* HERE IS NEW LOCATION OF CHECK ON AIDLPRIM
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BH    SATH020             SKIP IF A PRIMARY AUTH ID EXISTS
* END OF MOVED CODE - CHECK ON AIDLPRIM
         L     R7,ASCBJBNS         GET ADDRESS OF LOGON ID
         MVC   AIDLPRIM,0(R7)      MAKE IT THE PRIMARY AUTH ID
         B     SATH019             TO END OF THIS ROUTINE
SATH010  DS    0H                  NOT TSO, BUT BATCH OR STC SPACE
         L     R6,PSATOLD-PSA      CURRENT TCB ADDRESS
* START OF CODE ADDED TO SUPPORT TCP/IP NDB MULTI DB2 SUBSYSTEM FUNCT
         USING TCB,R6
         L     R7,TCBSENV          CURRENT ACEE ADDRESS
         USING ACEE,R7
         CLC   ACEEACEE,EYEACEE    IS THIS AN ACEE?
         BNE   SATH015             NO, GO USE JOB USER ID
         L     R14,TCBJSCB         ADDRESS OF JSCB
         USING IEZJSCB,R14
         CLC   JSCBPGMN,=CL8'PORTCLNT' IS IT NDB SERVER AS ?
         BNE   SATH015             NO, GO USE JOB USER ID
         MVC   AIDLPRIM,ACEEUSRI   MOVE USER ID INTO AIDL PRIMARY
         MVC   AIDLACEE,TCBSENV    MOVE ACEE POINTER INTO AIDL
         B     SATH020
         DROP  R6,R7,R14
SATH015  DS    0H
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BH    SATH020             SKIP IF A PRIMARY AUTH ID EXISTS
* END OF CODE ADDED TO SUPPORT TCP/IP NDB MULTI DB2 SUBSYSTEM FUNCTION
         L     R7,TCBJSCB-TCB(,R6) CURRENT JSCB ADDRESS
         L     R5,JSCBJCT-IEZJSCB(,R7) CURRENT JCT ADDRESS
         LA    R5,X'10'(,R5)       ADJUST FOR CORRECT DSECT MAPPING
         MVC   AIDLPRIM(7),JCTUSER-INJMJCT(R5)  COPY JOB USER ID
```

```
         MVI   AIDLPRIM+7,BLANK    ASSURE BLANK PADDING
SATH019  DS    0H                 END OF ROUTINE
         EJECT
⋮
```

Step 4: Updating the PORTS cataloged procedure

The PORTS procedure starts the NDB Port Manager. Update the PORTS cataloged procedure by copying the sample in *hlq*.SEZAINST(PORTSPRC) to your system or recognized PROCLIB and modifying it to suit your local installation. Change the DD statements, as required. Refer to *z/OS Communications Server: IP Configuration Reference* for the PORTS procedure.

Note: You must start PORTS before you start PORTC.

Step 5: Updating the PORTC cataloged procedure

The PORTC procedure starts the NDB Port Client and the NDB Servers. Update the PORTC cataloged procedure by copying the sample in *hlq*.SEZAINST(PORTCPRC) to your system or recognized PROCLIB and modifying it to suit your local installation. Refer to the *z/OS Communications Server: IP Configuration Reference* for a copy of PORTC.

Running multiple PORTC procedures

Currently the NDB Port Manager is able to manage up to 100 NDB Servers. Each PORTC procedure is able to start up to 20 NDB servers. The number of NDB Servers that can be started in a single address space depends on a number of factors, such as how large a region size can be specified and the size of the catalog work area. These factors may limit the number of NDB Servers able to start in a single address space to less than the maximum of 20.

In order to reach the desired number of NDB Servers, you can run multiple PORTC procedures. When starting a PORTC procedure, one of the parameters specified is DB2SSID. See *z/OS Communications Server: IP Configuration Reference* for more information on the DBSSID parameter. This parameter indicates which DB2 subsystem all NDB servers running in that address space will access. Each started PORTC procedure may contain a different value for the DBSSID parameter; that is, when starting multiple PORTC procedures, each procedure may point to the same or different DB2 subsystems. To do this:

1. Copy your customized PORTC cataloged procedure to another data set or PDS member.
2. Change the name in the first statement (PROC statement). For example if your original PORTC procedure starts with //PORTC PROC, use //PORTC1 PROC and //PORTC2 PROC for the other procedures.
3. Ensure that:
   - HOMEID parameter settings are the same for all PORTC procedures.
   - When the NUMSRV parameter settings for all the PORTC procedures are added together, the total does not exceed 100.
   - The NUMSRV value for any given PORTC does not exceed 20.
4. Start the new PORTC procedures.

Step 6: Creating the NDB clients

z/OS CS provides NDB sample client code that can be compiled and run in a variety of environments. NDB clients can be created on VM, MVS, or on DOS,

OS/2, AIX on RS/6000, or SUN UNIX workstations. You can find the NDB client source code in the *hlq*.SEZAINST data set, unless otherwise noted. They are written in the C language.

The process to create a client is similar in each case. Instructions for creating a client in each of the supported environments are specified. The MVS client code shipped with TCP/IP is ready to use without modification but you can modify it, if necessary, to suit your environment.

Note: To build an NDB client, you need access to the C runtime libraries and the TCP/IP RPC libraries. The TCP/IP products for the client platforms must be installed on those platforms before you build the NDB clients. In addition, for the DOS and OS/2 platforms, you must also install the TCP/IP Programmer's Toolkit.

Creating an NDB client in the AIX environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Issue the command:

```
cc -o ndbclnt ndbmain.c ndbc.c ndbclt.c ndbout.c ndbpclt.c ndbx.c
```

Notes:

1. This version of the NDB sample client code was produced and tested on AIX Version 3 Release 2 for RISC System/6000.
2. The file NDBRPCFC (ndbrpcf.c) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new copy before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See *TCP/IP AIX Programmer's Reference* for more information on the RPCGEN process.

Creating an NDB client in the SUN UNIX environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Issue the command:

```
cc -o ndbclnt -DNOPROTO ndbmain.c ndbc.c ndbclt.c ndbout.c ndbx.c
   ndbpclt.c
```

Notes:

1. This version of the NDB sample client code was produced and tested on SUN OS Version 4 Release 1 Modification 1.
2. The file NDBRPCFC (ndbrpcf.c) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the MVS TCP/IP library, do the following to generate a new copy before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

Refer to *SUN Network Programming* for more information.

Creating an NDB client in the OS/2 environment

1. Bring the following C source programs down to your workstation:

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBRPCFC	ndbrpcf.c
NDBXC	ndbx.c

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBCLTH	ndbclt.h

NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

3. Bring the following OS/2 files down to your workstation:

Program	Rename to
NDBCLDEF	ndbclnt.def
NDBOS2M	ndbos2.mak

   NDBOS2M has been stored as a binary file on MVS and, as such, cannot be read on the MVS host. When using FTP to move this file from MVS to OS/2, make sure the transfer type used is BINARY. The OS/2 makefile (NDBOS2M) is in the SEZAXLD2 data set.

4. Issue the command:

```
nmake -f ndbos2.mak
```

Notes:

1. This version of the NDB sample client code was produced and tested on OS/2 Warp Version 3.0, TCP/IP for OS/2 Version 3.0, including the TCP/IP for OS/2 Programmer's Toolkit, and IBM C Set ++ Version 2.0.
2. If the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the z/OS TCP/IP library, do the following to generate a new one before compiling the NDB sample client code.
   a. Bring the following X (RPC input) file down to your workstation:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

See *z/OS Communications Server: IP Programmer's Reference* and *CICS Communicating with CICS OS/2* for more information.

Creating an NDB client in the DOS environment

1. Bring the following C source programs down to your workstation

Program	Rename to
NDBCC	ndbc.c
NDBCLTC	ndbclt.c
NDBMAINC	ndbmain.c
NDBOUTC	ndbout.c
NDBPCLTC	ndbpclt.c
NDBRPCFC	ndbrpcf.c
NDBXC	ndbx.c

   When using FTP to move these files from MVS to DOS, make sure the transfer type used is ASCII.

2. Bring the following H (header) files down to your workstation:

Program	Rename to
NDBCLTH	ndbclt.h
NDBGLOBH	ndbglob.h
NDBH	ndb.h
NDBOUTH	ndbout.h
NDBPCLTH	ndbpclt.h
NDBRPCFH	ndbrpcf.h

When using FTP to move these files from MVS to DOS, make sure the transfer type used is ASCII.

3. Bring the following DOS makefile down to your workstation:

Program	Rename to
NDBDOSM	ndbdos.mak

NDBDOSM has been stored as an binary file on MVS and, as such, cannot be read on the MVS host. When using FTP to move this file from MVS to DOS, make sure the transfer type used is BINARY. The DOS makefile (NDBDOSM) is in the SEZAXLD2 data set.

4. Issue the command:

```
nmake -f ndbdos.mak
```

Notes:

1. This version of the NDB sample client code was produced and tested on IBM DOS Version 6.0, TCP/IP for DOS Version 2.1.1.4, including the TCP/IP for DOS Programmer's Toolkit [2], Microsoft Windows Version 3.1 and Microsoft C/C++ Version 7.0.

   The TCP/IP for DOS Programmer's Toolkit Version 2.1.1.4 contains a fix to the RPC library required to run the NDB DOS client.

2. TCP/IP for DOS does not support RPC server functions because it does not have a PORTMAPPER. Therefore, if the NDBH (ndb.h) file, an output of the RPCGEN process, is erased or corrupted and cannot be retrieved from the z/OS TCP/IP library, do the following to generate a new one on a workstation or mainframe that does support RPC server functions before compiling the NDB sample client code.

   a. Bring the following X (RPC input) file down to your workstation or mainframe:

Program	Rename to
NDBXX	ndb.x

   b. Issue the command:

```
rpcgen -h -o ndb.h ndb.x
```

   c. Transfer the resulting NDB.H file to your DOS workstation

See *TCP/IP DOS Programmer's Reference*, and the programmer's reference manuals for the workstation or the mainframe being used for RPCGEN for more information.

Creating an NDB client in the VM environment

1. Bring the following C source programs to your VM user ID:

Program	Rename to
NDBCC	NDBC C
NDBCLTC	NDBCLT C
NDBMAINC	NDBMAIN C
NDBOUTC	NDBOUT C
NDBPCLTC	NDBPCLT C
NDBXC	NDBX C

2. Bring the following H (header) files to your VM user ID:

Program	Rename to

2. The TCP/IP for DOS Programmer's Toolkit Version 2.1.1.4 contains a fix to the RPC library that is needed for the NDB DOS client to run.

NDBCLTH	NDBCLT H
NDBGLOBH	NDBGLOB H
NDBH	NDB H
NDBOUTH	NDBOUT H
NDBPCLTH	NDBPCLT H
NDBRPCFH	NDBRPCF H

3. Bring the following Assembler file to your VM user ID:

Program	Rename to
NDBVMPWA	NDBVMPW ASSEMBLE

4. Bring the following VM REXX EXEC to your VM user ID:

Program	Rename to
NDBCLBD	NDBCLBD EXEC

5. Execute the EXEC by entering:

```
ndbclbd
```

Notes:

1. This version of the NDB Sample Client code was produced and tested on VM/ESA® Version 1 Release 2 using CMS Level 9, running in XA mode and in ESA mode.
2. The file NDBRPCFC (NDBRPCF C) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (NDB H) file, an output of the RPCGEN process, should be erased or corrupted and cannot be retrieved from the MVS TCP/IP library, you will need to generate a new one before you can successfully compile the NDB sample client code. To do this:
   a. Bring the following X (RPC input) file to your VM user ID:

Program	Rename to
NDBXX	NDB X

   b. Issue the command:

```
rpcgen -h -o ndb h ndb x
```

See *TCP/IP for VM Programmer's Reference* for further information.

Creating an NDB client in the MVS environment

1. Copy the following C source programs to a PDS:

Program	Rename to
NDBCC	NDBC
NDBCLTC	NDBCLT
NDBMAINC	NDBMAIN
NDBOUTC	NDBOUT
NDBPCLTC	NDBPCLT
NDBXC	NDBX

2. Copy the following H (header) files to a PDS:

Program	Rename to
NDBCLTH	NDBCLT
NDBGLOBH	NDBGLOB
NDBH	NDB
NDBOUTH	NDBOUT
NDBPCLTH	NDBPCLT
NDBRPCFH	NDBRPCF

3. Copy the following sample JCL to a data set or PDS and customize it following the instructions found in the sample:
   NDBCLMVS
4. Execute the JCL.

Notes:

1. This version of the NDB Sample Client code was produced and tested on MVS/ESA Version 4 Release 2 Modification 2.
2. The file NDBRPCFC (NDBRPCF C) is not necessary unless the level of RPC being used does not support the RPC clnt_create function.
3. If the NDBH (NDB H) file, an output of the RPCGEN process, should be erased or corrupted and cannot be retrieved from the MVS TCP/IP library, you will need to generate a new one before you can successfully build the NDB sample client. To do this:
   a. Copy the following X (RPC input) file to a data set:

Program	Rename to
NDBXX	NDB.X

   b. Issue the command:

      ```
      rpcgen -h -o ndb.h ndb.x
      ```

See the *z/OS Communications Server: IP Programmer's Reference* for further information.

Starting NDB

Follow these steps to start NDB:

1. Ensure that the Portmapper is up and running. For more information, see "Configuring the PORTMAP address space" on page 653.
2. Run the PORTSPRC procedure to start the NDB Port Manager.
3. After PORTSPRC has started, run the PORTCPRC procedure to start the NDB Port Client and NDB Servers. Specify the start parameters as required.
4. Ensure that the required DB2 subsystem is running.
5. Invoke the NDB client.

Chapter 17. Mail servers

Configuring the SMTP server

Before you configure...

Read “Understanding search orders of configuration information” on page 18. It covers important information about data set naming and search sequences.

Note: Before configuring the SMTP server, it is assumed that the necessary SYS1.PARMLIB changes have been made. Consult the Program Directory for current information about the storage estimates for this version. The Program Directory also contains information about customization of certain SYS1.PARMLIB members, which must be completed before the initial program load (IPL) for the MVS image.

This chapter describes how to configure the Simple Mail Transfer Protocol (SMTP) server. There is also a section about operating the SMTP server.

The (SMTP or LPD) server uses the Pascal socket API, so VMCF must be started for the server to successfully initialize. If VMCF is not started, message EZY1980E will be issued and the server will terminate.

If you have specified PROFILE NOINTERCOM in your TSO user ID’s profile, then there are some SMTP server messages that you will not receive.

Checklist for working within the SMTP environment

1. SMTP needs to interface with JES utilities to create, read, write and purge data from the JES spool. JES exit programs might interfere with SMTP functioning properly.
2. JES initialization parameters must be set up correctly so mail can be sent to SMTP and so that local mail can be placed on the JES spool for local users.
3. Because SMTP needs authority to create, read, write and purge data on the JES spool, any security programs such as RACF, protecting JES spool access, must have the SMTP started task name in their definitions of authorized users.
4. DASD management is important to have SMTP run properly, it is recommended that SMTP have its own dedicated volumes. The MAILFILEVOLUME statement may be used to specify a particular volume where newly allocated SMTP data sets reside. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information on this parameter.
5. SMTP is a heavy user of data set I/O functions. It creates data sets for every piece of mail it processes. There are two data sets associated with each piece of mail:
   a. SMTPhlq.*.ADDRBLOK (control file)
   b. SMTPhlq.*.NOTE (message content)

   The SMTP high level qualifier (SMTPhlq) is configured in the SMTP configuration data set using the MAILFILEDSPREFIX statement. When SMTP is executing, SMTP must have exclusive access to the data sets it has created to work properly.

   To avoid contention, applications that manage DASD should only be run when SMTP is NOT active or exclude the SMTP data sets or exclude the volume(s)

on which they resided from their processing. If contention occurs you may see EZA5335E messages repeating. Also the SMTP high level qualifier can be used to exclude the SMTP data sets if necessary.

Configuration process

Steps to configure SMTP:

1. Specify AUTOLOG and PORT statements in the *hlq*.PROFILE.TCPIP data set.
2. Update the SMTP cataloged procedure *hlq*.SEZAINST(SMTPPROC).
3. Customize the SMTPNOTE CLIST and modify PARMLIB members.
4. Customize the SMTP mail headers (optional).
5. Set up a TCP-to-NJE mail gateway (optional).
6. Specify configuration statements in the SMTP configuration data set.
7. Create an SMTP security table (optional).
8. Enable SMTP domain name resolution.
9. Enable sending messages to SMTP users and users on an IP Network.
10. Optionally, design SMTP exit to inspect and filter unwanted mail (*spam*)

Step 1: Specify AUTOLOG and PORT statements in hlq.PROFILE.TCPIP

If you want the SMTP server to start automatically when the TCPIP address space is started, include the name of the member containing the SMTP cataloged procedure in the AUTOLOG statement of the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  SMTP
ENDAUTOLOG
```

To ensure that port TCP 25 is reserved for SMTP, verify that the name of the member containing the SMTP cataloged procedure has been added to the PORT statement in *hlq*.PROFILE.TCPIP.

```
PORT
  25 TCP SMTP
```

For more information on the AUTOLOG and PORT statements, see *z/OS Communications Server: IP Configuration Reference*.

HOME statement in the *hlq*.PROFILE.TCPIP data set consideration: SMTP allows a maximum of 255 entries on the HOME statement.

Step 2: Update the SMTP cataloged procedure

Update the SMTP cataloged procedure by copying the sample in *hlq*.SEZAINST(SMTPPROC) to your system or recognized PROCLIB. Specify SMTP parameters, and change the data set as required to suit your local configuration. Refer to *z/OS Communications Server: IP Configuration Reference* for more detailed information about the procedure.

Note: SMTP does not support HFS files.

Step 3: Customize the system CLIST and modify PARMLIB data sets

You must copy and customize the SMTPNOTE CLIST on every system where users will be able to send mail with the SMTPNOTE command. This includes TCP/IP nodes and each NJE node that sends mail through SMTP on a remote gateway node. SMTPNOTE uses the TSO transmit (XMIT) command to interface with SMTP.

Copy the SMTPNOTE member of the *hlq*.SEZAINST data set into the system CLIST data set. Since the *hlq*.SEZAINST data set is in a fixed format, the SMTPNOTE member may be truncated if your system CLIST library is not in a fixed format.

You should customize the following variables in the SMTPNOTE CLIST:

DDNAME
The DDNAME that SMTPNOTE will use to allocate the input data set. The allocation is done to allow shared access to the data set. The default value is set to EZBSMTPN and should only be changed if this value will cause a conflict on your system.

HOSTNAME
The name of the system on which this CLIST is installed. Typically, the name is the NJE node name of this system.

SMTPNODE
The NJE node on which the SMTP server runs. Typically, HOSTNAME and SMTPNODE have the same value. When SMTPNODE is used on an NJE network in conjunction with a TCP-to-NJE gateway, the value of this parameter is the NJE node name of that gateway.

SMTPJOB
The name of the address space in which SMTP runs at SMTPNODE. Usually this is SMTP.

TEMPDSN
The name of the temporary data set used to store the contents of the note being created. This can be any arbitrary data set name that ends with the low-level qualifier, TEXT. Do not use a fully qualified name. If you do not fully qualify the name (no quotes), the data set name will be prefixed by the *userid*. If you enclose the name in single quotes, several users can use this temporary data set.

TIMEZONE
The time zone for your system. This will appear in the ″Date:″ stamp of the RFC 822 header. See RFC 822 for valid time zone formats.

ATSIGN
Some foreign languages need to use a different character to represent the @ symbol. This input symbol is a single byte representation of the @ symbol in their national language code page.

You should also modify the following PARMLIB members:

IEFSSNxx
The IEFSSNxx member may be modified in one of the following two ways:

- The following lines may be included:

```
TNF,MVPTSSI
VMCF,MVPXSSI, nodename
```

  where *nodename* is the NJE node name. The NJE node name, *nodename*, must be the same as the *hostname* and the *smtpnode* in the SMTPNOTE CLIST.
- If you are using restartable VMCF, you must make changes to IEFSSxx members in the SYS1.PARMLIB data set.

  For introductory information on restartable VMCF, refer to “Step 3: Configure VMCF and TNF” on page 72. For the MVS system changes

required for restartable VMCF, refer to the TCP/IP for MVS Program Directory. For information on VMCF commands, refer to *z/OS Communications Server: IP Diagnosis*.

Note: You should define the SystemName in the IEFSSNxx PARMLIB member to be the same as your JES2 or JES3 (NJE) nodename. This is required for correct delivery of SMTP mail. For example, if the following line is coded in your TCPIP.DATA data set:

```
HOSTNAME P390
```

you need to code NAME=P390 in your IEFSSNxx PARMLIB member. As an alternative, instead of using the IEFSSNxx parmlib member to specify the JES node, you can use the keyword NJENODENAME within your SMTP configuration to a valid NJE node. For more information, see NJENODENAME.

IKJTSOxx
The TRANSREC statement must contain the correct nodename.

Step 4: Customize the SMTP mail headers (Optional)

Electronic mail has a standardized syntax for text messages that are sent across networks. The standard syntax is described in RFC 822. Messages have an envelope and contents. Envelopes contain all necessary information to accomplish transmission and delivery of the message content. The fields within the envelope are in a standard format.

In most cases, the mail header defaults are adequate. If it is necessary for you to change them, you can use the REWRITE822HEADER statement in the SMTP configuration data set to control the way SMTP performs a rewrite of the RFC 822 mail headers. Mail headers are passed from the local system, or NJE network, to the TCP network. Mail headers passing from the TCP network to the local system or NJE network are not affected. Only the addresses under certain RFC 822 header fields can be subject to the header rewriting rules.

The header fields affected by the REWRITE822HEADER statement are:

Field **Description**

From The identity of the person sending the message.

Resent-From
Indicates the person that forwarded the message.

Reply-To
Provides a mechanism for indicating any mailboxes to which responses are to be sent.

Resent-Reply-To
Indicates the person to whom you should forward the reply.

Return-Path
This field is added by the mail transport service at the time of final delivery. It contains definitive information about the address and route back to the originator of the message.

Sender
The authenticated identity of the agent that sent the message. An agent can be a person, system, or process.

Resent-Sender
The authenticated identity of the agent that has resent the message.

To Contains the identity of the primary recipient of the message.

Cc Contains the identity of the secondary (informational) recipients of the message.

Bcc Contains the identity of additional recipients of the message. The contents of this field are not included in copies sent to the primary and secondary recipients of the message but are included in the author's copy.

The SMTP rules data set: You can override the default rules for header addresses by creating an SMTP rules data set. This allows you to customize the address transformations to the needs of a particular site. If you are customizing SMTP mail headers, this task is required.

The SMTP rules data set is pointed to by the //SMTPRULE DD statement in the SMTP cataloged procedure. The SMTP rules data set consists of:

Field definition
Contains the names of all header fields whose addresses are to be rewritten.

Rules definition
Contains the rewrite rules for the header fields.

Statement syntax: In creating the SMTP rules data set you must use the following syntax conventions:

- The data set statements are free-format. Tokens can be separated by an arbitrary number of spaces, and statements can span an arbitrary number of lines. However, you must end every statement with a semicolon (;).
- A character string appearing within single quotation marks ('...') is not case-sensitive. For example, 'abc' represents 'abc', 'Abc', 'ABC', and so forth. The use of character strings is illustrated in the following sections.
- A character string appearing within double quotation marks ("...") is case-sensitive. For example, "abc" only represents "abc". It does not represent "Abc", "ABC", and so forth.

  Special characters, such as @ and % are treated the same whether enclosed by single quotation marks or double quotation marks.
- Double-hyphens ("--") are used to begin a comment. The comment extends to the end of the line.

The following sections describe the components of the SMTP rules data set.
- Format of the field definition section
- Format of the rules definition section

Format of the field definition section: The field definition section is the first section in any SMTP rules data set. It defines any applicable alias fields, and it is introduced by the following heading:

```
Field Definition Section
```

This section allows similar fields to be grouped under an alias or common name. This name, or alias, is used to represent the field list. You can define an arbitrary number of aliases representing a set of field lists.

An alias name can be any alphanumeric sequence of characters that is not a predefined keyword within the SMTP rules (see the following). However, the alias name `DefaultFields` is treated specially by the SMTP configuration interpreter. If

DefaultFields is defined, and if a rule is written that does not specify an associated field alias, the rules interpreter assumes that DefaultFields is the associated field alias.

The alias definition within this section is of the following form:

```
alias_name = alias_definition; optional comment
```

where *alias_name* is the name of the alias and *alias_definition* is an expression describing which fields are to be grouped under this alias. This expression can be as simple as a single field name. For example:

```
MyAlias = 'To';
```

The aliases can be a list or set of field names. The field names ***To, From, Cc***, and ***Bcc***, in the following example are part of a set of field names referenced by the alias MyAlias.

```
MyAlias = 'To' 'From' 'Cc' 'Bcc' ; -- first list of fields
```

You can combine field names and previously defined aliases to create a new alias. In the following example, the set of field names defined as MyAlias and the field names in the new alias YourAlias are combined to form a third set. The new alias TheirAlias is the union of both aliases and contains the fields of MyAlias and YourAlias.

```
MyAlias    = 'To' 'From' 'Cc' 'Bcc';
YourAlias  = 'Errors-To' 'Warnings-To';
TheirAlias =  MyAlias YourAlias;
```

In the previous example, TheirAlias is an alias that represents the following fields:

```
TheirAlias:  'To' 'From' 'Cc' 'Bcc' 'Errors-To' 'Warnings-To'
```

You can perform the following operations on set members of the alias to create a subset of the initial alias:

- Union operations
- Difference operations
- Intersection operations

Union and difference operations: Certain field names can be added to or omitted from a new alias of field names by using a minus sign to omit set members and an optional plus sign to include another field name. In the mathematics of sets, when you add together 2 or more sets, they form a union. When set members are omitted, the remaining set is created by the difference operation. In the following example HerAlias and HisAlias are defined. The alias HisAlias is created from the union of TheirAlias, HerAlias, and the omission of Warning-To and Bcc from the sets:

```
HerAlias   = 'Reply-To' 'Sender';
HisAlias   = TheirAlias - 'Warnings-To' - 'Bcc' + HerAlias;
```

In the previous example, HisAlias is an alias that represents the following fields:

```
HisAlias:  'To' 'From' 'Cc' 'Errors-To' 'Reply-To' 'Sender'
```

Intersection operations: A field definition can include an intersection operation. When the intersection operation is applied to two field expressions, the resulting set contains the fields common to both. In the following example, MyAlias and YourAlias are defined. The alias OurAlias is created from the intersection of MyAlias and YourAlias. The asterisk (*) is the intersection operator.

```
MyAlias    = 'Bcc' 'Cc' 'From' 'Reply-To';
YourAlias  = 'Resent-From' 'Cc' 'Sender' 'To' 'Bcc';
OurAlias   = MyAlias * YourAlias; -- the intersection
```

In the previous example, `OurAlias` represents the following fields:

```
OurAlias:  'Bcc' 'Cc'
```

In the following complex example `TheirAlias` is created from the intersection of `YourAlias` with the sum of `MyAlias` plus `Resent-From`:

```
TheirAlias = (MyAlias + 'Resent-From') * YourAlias;
```

In the previous example, `TheirAlias` represents the following fields:

```
TheirAlias: 'Bcc' 'Cc' 'Resent-From'
```

The parentheses within the definition of `TheirAlias` perform the same functions as in algebra. Field expressions are evaluated from left to right, but the intersection operation has greater priority than union and difference operations. If parentheses were not used in the definition of `TheirAlias`, the result would be:

```
TheirAlias: 'Bcc' 'Cc' 'From' 'Reply-To' 'Resent-From'
```

Format of the rule definition section: The rule definition section is the next section in any SMTP RULES data set. It contains the header rewriting rules that define the intended address transformations, and it is introduced by the following heading:

```
Rule Definition Section
```

The basic form of a rewrite rule is:

alias :*before-address-pattern* => *after-address-pattern*;

where the alias name *alias* is an optional name representing the fields for which the rule is applicable. If the alias name *alias* : is omitted from this part of the rules, then `DefaultFields` is assumed.

The sequence of tokens that define how a particular type of address is to be recognized is the *before-address-pattern* portion of the rules definition. The sequence of tokens that define how the address is to appear after the address has been rewritten is the *after-address-pattern* portion of the rules definition. The following example is the rule for converting host names:

```
A '@' NJEHostName => A '@' TCPHostName; -- convert host names
```

In the previous example, `A '@' NJEHostName` is the *before-address-pattern* portion of this rule, and `A '@' TCPHostName` is the *after-address-pattern* portion. This rule specifies that the address to be rewritten has an arbitrary local name (`A`), an at sign (@), and the NJE host name (`NJEHostName`) of the current site. The rule also specifies that the rewritten address must contain the same arbitrary local name (`A`), an at sign, and the current site's TCP host name `TCPHostName`.

SMTP rules syntax conventions: Use the following syntax convention when writing SMTP rules:

- Some keywords have special meaning to the rules interpreter. For example, `NJEHostName` keyword means the NJE host name of the present system, and `TCPHostName` keyword means the TCP host name of the present system. For more information about valid keywords see "Predefined keywords within the SMTP rules" on page 677. Some keywords, such as `TCPHostName`, have single values. Other keywords, such as `AltTCPHostName` and `AnyDomainName`, can have many possible values. To avoid ambiguity, any keyword that can have multiple

values, and is used in the *after-address-pattern* of a given rule, must appear exactly once within the *before-address-pattern* of that rule. The following rule example shows a valid syntax:

```
A '@' AltTCPHostName '.' AltTCPHostName =>
   A '%' TCPHostName '@' TCPHostName;
```

The following two rules have incorrect syntax because the first keyword `AltTCPHostName` must be rewritten to a keyword with specific values. The `AltTCPHostName` is attempting to be rewritten to the same `AltTCPHostName` but with unknown values, which is not valid.

```
A '@' AltTCPHostName '.' AltTCPHostName =>
   A '%' AltTCPHostName '@' TCPHostName;

A '@' TCPHostName => A '@' AltTCPHostName;
```

Any rule whose *before-address-pattern* includes a keyword that has a null value is ignored during the header rewriting. Thus, if there is no `AltNJEDomain` defined in the system configuration data set, no rule that includes `AltNJEDomain` in the *before-address-pattern* is considered during the header rewriting.

- Alphanumeric identifiers that are not within single or double quotation marks, and that are not predefined keywords, are considered wildcards in the rule statement. Wildcards represent an arbitrary (non-null) sequence of characters. The identifier `A`, in the previous rule example, is a wildcard. Thus, if `host` were the NJE host name for the current site, and if `tcphost` were the TCP host name for the current site, the previous rule example recognizes `abc@host` and `d@host` as candidates for address rewriting, and rewrites them as `abc@tcphost` and `d@tcphost` respectively. To avoid ambiguity, within the *before-address-pattern* of a given rule, no two wildcards are allowed in a row, and the same wildcard cannot be used more than once. The following rules have valid syntax:

  ```
  A '@' B TCPHostName      => A '%' B '@' TCPHostName;
  A '%' B '@' NJEHostName => A B '@' TCPHostName;
  ```

  The following rules have incorrect syntax because the first rule has 2 wildcards in a row `A` and `B`. The second rule has the same wildcard `A` repeated:

  ```
  A B '@' TCPHostName      => A A '%' B '@' TCPHostName;
  A '%' A '@' NJEHostName => A '@' TCPHostName;
  ```

- A character string appearing within single or double quotation marks tells the rules interpreter where a particular string is to appear within a header address. In the previous rule example, the `'@'` string in the *before-address-pattern* tells the rules interpreter that an at-sign (@) must appear between the arbitrary character string and the NJE host name. The `'@'` string in the *after-address-pattern* tells the rules interpreter that the address must be rewritten so an at-sign appears between the arbitrary string and the TCP host name. As previously mentioned, single quotation marks denote strings that are not case-sensitive, and double quotation marks denote case-sensitive strings.
- The character sequence `"=>"`, with no spaces between the characters, separates the *before-address-pattern* from the *after-address-pattern*.
- The order in which the rules are specified is important; the first rule encountered whose *before-address-pattern* matches the current address is the rule to dictate the address transformation. Once a matching rule has been found for an address, no other rule is considered.

In addition to the rules themselves, there is the capability for some simple logic to decide at system configuration time which rules within the data set should become active. These conditions are specified in the form of an IF-THEN-ELSE statement, as shown in the following example:

```
IF cond THEN
   statement list
ELSE
   statement list
ENDIF
```

A statement list can consist of any number of rules or nested IF statements, or both. Each IF statement, regardless of whether it is nested, must be ended by an ENDIF keyword. As with IF statements in other programming languages, the ELSE clause is optional.

There are only two conditions recognized by an IF statement:
1. IF *predefined keyword* = '*character string*' THEN ... ENDIF
2. IF *predefined keyword* CONTAINS '*character string*' THEN ... ENDIF

The conditional operators = and CONTAINS can be prefixed by the word NOT to invert the conditions.

The *predefined keyword* must be a keyword that resolves to a single value at system configuration time. The character string in the first condition can be null. A character string cannot span more than one line.

The following example shows the use of IF statements.

```
IF NJEDomain = " THEN
   A '@' AnyNJEHostName => A '%' AnyNJEHostName '@' TCPHostName;
ELSE
   A '@' NJEHostName '.' NJEDomain    => A '@' TCPHostName;
   A '@' NJEHostName '.' AltNJEDomain => A '@' TCPHostName;
   IF NJEDomain CONTAINS '.' THEN
      A '@' AnyNJEHostName                   =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '@' AnyNJEHostName '.' NJEDomain;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '@' AnyNJEHostName '.' NJEDomain;
   ELSE
      A '@' AnyNJEHostName                    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' NJEDomain    =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
      A '@' AnyNJEHostName '.' AltNJEDomain =>
         A '%' AnyNJEHostName '.' NJEDomain '@' TCPHostName;
   ENDIF
ENDIF
```

Predefined keywords within the SMTP rules: The following predefined keywords can be used to define the header rewriting rules:

AltNJEDomain
: Matches the alternative domain name of the NJE network as defined by the ALTNJEDOMAIN statement in the SMTP configuration data set.

AltTCPHostName
: Matches any alternative TCP host name of the system, as defined by ALTTCPHOSTNAME statements in the SMTP configuration data set.

AnyDomainName
Matches any fully qualified domain name. Any host name with a period (.) is considered to be a fully qualified domain name.

AnyNJEHostName
Matches any (unqualified) NJE host name defined in the SMTPNJE.HOSTINFO data set.

NJEDomain
Matches the domain name of the NJE network as defined by the NJEDOMAIN statement in the SMTP configuration data set.

NJEHostName
Matches the NJE host name of the system.

SecureNickAddr
Matches an address of the form *NJE_user_id@NJE_node_id*, where *NJE_user_id*, and *NJE_node_id* are defined with a nickname in the SMTP security data set.

Note: This only matches user and node IDs that are defined with nicknames.

When SecureNickAddr is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickName with the corresponding nickname. This allows SecureNickName to be specified in the *after-address-pattern*.

SecureNickName
Matches a nickname defined in the SMTP security data set. When SecureNickName is specified in the *before-address-pattern* of a rule, SMTP automatically associates the keyword SecureNickAddr with the corresponding *NJE_user_id@NJE_node_id*. This allows SecureNickAddr to be specified in the *after-address-pattern*.

ShortTCPHostName
Matches the first portion of the TCP host name of the system, as defined by the HOSTNAME statement in the TCPIP.DATA data set. For example, if the TCP host name was `mvs1.acme.com`, the value of ShortTCPHostName is `mvs1`.

TCPHostName
Matches the TCP host name of the system as defined by the concatenation of the HOSTNAME and DOMAINORIGIN statements in the TCPIP.DATA data set.

TCPHostNameDomain
Matches the domain portion of the TCP host name of the system as defined by the DOMAINORIGIN statement in the TCPIP.DATA data set. For example, if the TCP host name was `mvs1.acme.com`, the value of TCPHostNameDomain is `acme.com`.

The predefined keywords can consist of any combination of uppercase and lowercase characters; the rules interpreter does not distinguish between them.

The secure keywords are only valid when SMTP is configured to be a secure gateway.

Default SMTP rules: If the //SMTPRULE DD statement is not found, SMTP uses a default set of rules. The default set used depends on whether SMTP is configured as a secure gateway.

SMTP nonsecure gateway configuration defaults: If SMTP is not configured as a secure gateway, SMTP uses the following defaults:

```
FIELD DEFINITION SECTION
DEFAULTFIELDS = 'BCC' 'CC' 'FROM' 'REPLY-TO' 'RESENT-FROM'
                'RESENT-REPLY-TO' 'RESENT-SENDER' 'RETURN-PATH'
                'SENDER' 'TO';

RULE DEFINITION SECTION

A '@' NJEHOSTNAME => A '@' TCPHOSTNAME;

IF NJEDOMAIN = '' THEN
   A '@' ANYNJEHOSTNAME => A '%' ANYNJEHOSTNAME '@' TCPHOSTNAME;
ELSE
   A '@' NJEHOSTNAME '.' NJEDOMAIN    =>'@' TCPHOSTNAME;
   A '@' NJEHOSTNAME '.' ALTNJEDOMAIN =>'@' TCPHOSTNAME;
   IF NJEDOMAIN CONTAINS '.' THEN
      A '@' ANYNJEHOSTNAME                 =>
         A '@' ANYNJEHOSTNAME '.' NJEDOMAIN;
      A '@' ANYNJEHOSTNAME '.' NJEDOMAIN    =>
         A '@' ANYNJEHOSTNAME '.' NJEDOMAIN;
      A '@' ANYNJEHOSTNAME '.' ALTNJEDOMAIN =>
         A '@' ANYNJEHOSTNAME '.' NJEDOMAIN;
    ELSE
       A '@' ANYNJEHOSTNAME                  =>
          A '%' ANYNJEHOSTNAME '.' NJEDOMAIN '@' TCPHOSTNAME;
       A '@' ANYNJEHOSTNAME '.' NJEDOMAIN    =>
          A '%' ANYNJEHOSTNAME '.' NJEDOMAIN '@' TCPHOSTNAME;
       A '@' ANYNJEHOSTNAME '.' ALTNJEDOMAIN =>
          A '%' ANYNJEHOSTNAME '.' NJEDOMAIN '@' TCPHOSTNAME;
    ENDIF
 ENDIF

A '@' TCPHOSTNAME       => A '@' TCPHOSTNAME;
A '@' SHORTTCPHOSTNAME  => A '@' TCPHOSTNAME;
A '@' ALTTCPHOSTNAME    => A '@' TCPHOSTNAME;
A '@' ANYDOMAINNAME     => A '@' ANYDOMAINNAME;
A '@' B                 => A '@' B '.' TCPHOSTNAMEDOMAIN;
```

SMTP secure gateway configuration defaults: If SMTP is configured as a secure gateway, SMTP uses the following defaults:

```
FIELD DEFINITION SECTION
DEFAULTFIELDS = 'BCC' 'CC' 'FROM' 'REPLY-TO' 'RESENT-FROM'
                'RESENT-REPLY-TO' 'RESENT-SENDER' 'RETURN-PATH'
                'SENDER' 'TO';

RULE DEFINITION SECTION

SECURENICKADDR    => SECURENICKNAME '@' TCPHOSTNAME;
A '@' NJEHOSTNAME => A '@' TCPHOSTNAME;

IF NJEDOMAIN NOT = '' THEN
   SECURENICKADDR '.' NJEDOMAIN    => SECURENICKNAME '@' TCPHOSTNAME;
   SECURENICKADDR '.' ALTNJEDOMAIN => SECURENICKNAME '@' TCPHOSTNAME;
   A '@' NJEHOSTNAME '.' NJEDOMAIN    =>'@' TCPHOSTNAME;
   A '@' NJEHOSTNAME '.' ALTNJEDOMAIN =>'@' TCPHOSTNAME;
   IF NJEDOMAIN CONTAINS '.' THEN
      A '@' ANYNJEHOSTNAME                 =>
         A '@' ANYNJEHOSTNAME '.' NJEDOMAIN;
      A '@' ANYNJEHOSTNAME '.' NJEDOMAIN    =>
         A '@' ANYNJEHOSTNAME '.' NJEDOMAIN;
```

```
      A '@' ANYNJEHOSTNAME '.' ALTNJEDOMAIN =>
         A '@' ANYNJEHOSTNAME '.' NJEDOMAIN;
   ELSE
      A '@' ANYNJEHOSTNAME                 =>
         A '%' ANYNJEHOSTNAME '.' NJEDOMAIN '@' TCPHOSTNAME;
      A '@' ANYNJEHOSTNAME '.' NJEDOMAIN    =>
         A '%' ANYNJEHOSTNAME '.' NJEDOMAIN '@' TCPHOSTNAME;
      A '@' ANYNJEHOSTNAME '.' ALTNJEDOMAIN =>
         A '%' ANYNJEHOSTNAME '.' NJEDOMAIN '@' TCPHOSTNAME;
   ENDIF
ENDIF
A '@' TCPHOSTNAME       => A '@' TCPHOSTNAME;
A '@' SHORTTCPHOSTTNAME => A '@' TCPHOSTNAME;
A '@' ALTTCPHOSTNAME    => A '@' TCPHOSTNAME;
A '@' ANYDOMAINNAME     => A '@' ANYDOMAINNAME;
A '@' B                 => A '@' B '
```

Examples of header rewrite rules: The following examples show how the header rewriting rules affect an SMTP mail header. The example site is not a secure gateway and is configured as follows:

```
TCPHostName      = mvs1.acme.com
ShortTCPHostName = mvs1
AltTCPHostName   = seeds.acme.com
NJEHostName     = mvs1
NJEDomain       = acmenet
AltNJEDomain    = centralnet
```

Note that the above keywords are configured according to the definitions found in "Predefined keywords within the SMTP rules" on page 677 (for example, from TCPIP.DATA). In addition, assume that the following are known to be other NJE hosts:

```
bird
iron
```

Then the following header:

```
From: abc@mvs1 (Brendan Beeper)
To: Jenny Bird <def@bird>
Cc: ghi@iron.acmenet, j@mvs1,
 k@seeds.acme.com,
 Mailing List <owner@acmenet>,
 lmno@iron.centralnet
Subject: New Ore
```

is rewritten by the default header rewriting rules as:

```
From: abc@mvs1.acme.com (Brendan Beeper)
To: Jenny Bird <def%bird.acmenet@mvs1.acme.com>
Cc: ghi%iron.acmenet@mvs1.acme.com, j@mvs1.acme.com,
 k@mvs1.acme.com,
 Mailing List <owner%acmenet@mvs.acme.com>,
 lmno%iron.acmenet@mvs1.acme.com
Subject: New Ore
```

The next example deviates from the defaults listed in "Default SMTP rules" on page 679. On the configuration for nonsecure gateways, if you change the rule before the 2 ENDIFs to:

```
A '@' AnyNJEHostName '.' AltNJEDomain =>
   '<@' TCPHostName ':' A '@' AnyNJEHostName '.' NJEDomain '>';
```

then the last address in the Cc: field within our header is rewritten as:

```
Cc: <@mvs1.acme.com:lmno@iron.acmenet>
```

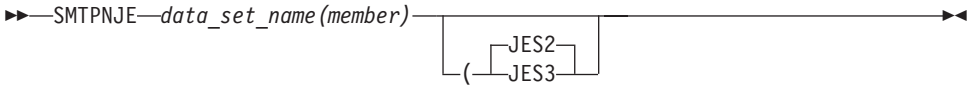
Note: Do not make the change shown in the previous example; it is intended only as a demonstration of the capabilities of the pattern-matching language.

Step 5: Set up a TCP-to-NJE mail gateway (Optional)

You can configure the SMTP server to run as a mail gateway between TCP network users and users located on a NJE network attached to the local host. This way NJE users can send mail or data sets to users on TCP hosts using SMTPNOTE. See *z/OS Communications Server: IP User's Guide and Commands* for more information about SMTPNOTE. For JES2, see *MVS/ESA JES2 Initialization and Tuning*, SC28-1038-1. For JES3, see *MVS/ESA JES3 Initialization and Tuning*, SC23-0073-2.

Follow these steps to set up your TCP-to-NJE mail gateway:

1. Add the GATEWAY statement to the SMTP configuration data set. Add other related statements, such as ALTDOMAIN, NJECLASS, NJEDOMAIN, and NJEFORMAT, as required by your configuration.
2. Issue the SMTPNJE command.



data_set_name(member)
The name of the input data set for SMTPNJE. It specifies the initialization data set of the JES2 or JES3 subsystem that is scanned for NJE nodes by SMTPNJE. The data set name is the same name as defined on ddname HASPPARM in your JES2 procedure or in the JES3IN ddname in your JES3 procedure.

member is the JES2PARM member that contains the NODE and DESTID entries for your installation.

(Required delimiter.

JES2 or JES3
Denotes whether the initialization data set being pointed to is for JES2 or JES3. If omitted, the default is JES2. For JES2, the SMTPNJE program scans for the keywords NODE and DESTID from which it extracts the information. For JES3, the keyword scanned for is NJERMT.

The SMTPNJE program creates the NJE host table data set called *user_id*.SMTPNJE.HOSTINFO. You can rename this data set and include the name of the data set on the SMTPNJE DD statement in the SMTP cataloged procedure. The //SMTPNJE DD statement is required.

3. Install the SMTP server (along with the TCPIP address space) on the gateway node. Use the GATEWAY, NJEDOMAIN, and NJEFORMAT statements in the configuration data set. Optionally, you can use either the RESTRICT or the SECURE statements to limit which users can use the gateway.

Step 6: Specify configuration statements in SMTP configuration data set

Copy the member *hlq*.SEZAINST(SMTPCONF) to your own SMTP configuration data set and modify it for your site using the SMTP configuration statements.

Note: If the SMTP configuration data set is a sequential data set, you cannot edit the data set while SMTP is running. If the data set is a PDS member, it can be edited while SMTP is running.

Summary of SMTP configuration statements: The SMTP configuration statements are summarized in Table 23.

Note: Refer to *z/OS Communications Server: IP Configuration Reference* for more information about these statements.

Table 23. Summary of SMTP configuration statements

Statement	***Description***
ALTNJEDOMAIN	Specifies an alternative domain name of the NJE network, if SMTP is running as a mail gateway.
ALTTCPHOSTNAME	Specifies an additional host name for the local host. Mail received for this host name is accepted and delivered locally.
ATSIGN	Enables SMTP to replace the @ symbol used in addressing strings.
BADSPOOLFILEID	Specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail.
CHECKSPOOLSIZE	Enables SMTP to check the size of the JES spool file prior to writing the data to the hlq.TEMP.NOTE file.
DBCS	Specifies that DBCS code conversion be performed on the mail.
DEBUG	Records all SMTP commands and replies.
FINISHOPEN	Specifies the SMTP wait time for connection.
GATEWAY	Specifies operation of SMTP as a gateway.
INACTIVE	Specifies the SMTP wait time before closing an inactive connection.
IPMAILERADDRESS	Specifies the IP address of an SMTP server that can resolve network addresses of unknown hosts.
LISTENONADDRESS	Allows you to restrict which IP address is used to receive mail on a multihomed system.
LOCALCLASS	Specifies the spool data set class for local mail delivery.
LOCALFORMAT	Specifies the spool data set format for local host mail delivery.
LOG	Directs SMTP to log all SMTP traffic.
MAILER	Specifies the address of the batch SMTP server that receives mail.
MAILFILEDSPREFIX	Specifies the prefix to add to mail data sets.
MAILFILESUNIT	Specifies the unit where SMTP mail data sets reside.
MAILFILEVOLUME	Specifies the volume where newly allocated SMTP data sets reside.
MAXMAILBYTES	Specifies the maximum size of mail that is accepted over a TCP connection.
NJECLASS	Specifies the spool data set class for mail delivered on an NJE network.
NJEDOMAIN	Specifies the domain name of the NJE network if SMTP functions as a gateway.
NJEFORMAT	Specifies the spool data set format for mail delivered on the NJE network.
NJENODENAME	Specifies the node name of the local JES2 or JES3 node for mail delivered on the NJE network.
NOLOG	Turns off the logging of mail transactions.
NOSOURCEROUTE	Enables SMTP to *not* generate source routing addressing strings on certain RFC 821 SMTP commands.
OUTBOUNDOPENLIMIT	Specifies a limit on the maximum number of simultaneous TCP connections over which SMTP actively delivers mail.
PORT	Specifies an alternative port number for the SMTP server during testing.

Table 23. Summary of SMTP configuration statements (continued)

Statement	***Description***
POSTMASTER	Specifies the address (or addresses) for mail addressed to the postmaster at the local host.
RCPTRESPONSEDELAY	Specifies how long the SMTP server delays responding to the RCPT commands.
RESOLVERRETRYINT	Specifies the number of minutes SMTP waits between attempts to resolve domain names.
RESOLVERUSAGE	Specifies whether SMTP will send queries to the domain name servers if they are configured in the TCPIP.DATA file.
RESTRICT	Specifies addresses of users who are not allowed to use SMTP mail services.
RETRYAGE	Specifies the number of days after which mail is returned as undeliverable.
RETRYINT	Specifies the number of minutes between attempts to send mail to an inactive TCP host.
REWRITE822HEADER	Prevents SMTP from rewriting RFC 822 headers with source routing.
SECURE	Specifies that SMTP operates as a secure mail gateway between TCP network sites and NJE network sites.
SMSGAUTHLIST	Specifies the addresses of users authorized to issue privileged SMTP SMSG commands.
SPOOLPOLLINTERVAL	Specifies the interval for SMTP to check the spool for incoming batch data sets.
TEMPERRORRETRIES	Specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem.
TIMEZONE	Sets the printable name of the local time zone.
WARNINGAGE	Specifies the number of days after which a copy of the mail is returned to the sender, indicating that the mail has so far been undeliverable and that SMTP will continue to retry delivery for RETRYAGE days.

Sample SMTP configuration data set (SMTPCONF): The sample SMTP Configuration data set can be found in *hlq*.SEZAINST(SMTPCONF). Refer to the *z/OS Communications Server: IP Configuration Reference* for more information on configuration data set parameters.

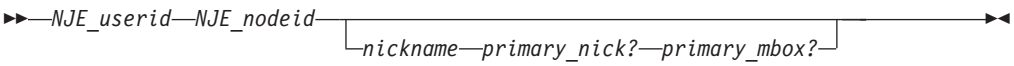
Step 7: Create an SMTP security table (Optional)

If you want to set up a secure TCP-to-NJE gateway, you need to:

- Include the SECURE statement in the SMTP configuration data set.
- Create a security data set that contains a list of NJE users who are authorized to use the gateway.
- Create a mailfiledsprefix.SECURITY.MEMO data set. The contents data set are sent to unauthorized NJE users whose mail is rejected. In this section, see “Rejected mail examples” on page 685 for sample contents of this data set. This data set must be defined as LRECL=255 and RECFM=VB. It will be dynamically allocated by SMTP when needed.

The SMTP security data set is pointed to by //SECTABLE DD statement. The security table data set must be allocated with LRECL=255 and RECFM=VB. Records whose first nonblank character is an asterisk (*) are treated as comments and are ignored.

Use the following format when creating the list of NJE users:



NJE_userid
The NJE user ID of the authorized user.

NJE_nodeid
The NJE node ID of the authorized user.

nickname
The name by which this user is known on the TCP side of the gateway. This name must not contain any special characters, such as < > ( ) [ ] \ . , ; : @ and ".

primary_nick?
Either Y or N. If Y is specified, then mail addressed to *nickname*`@smtp-gateway` is automatically forwarded to *NJE_userid* `at` *NJE_nodeid*. Each nickname can have only one *primary_nick?* record set to Y.

primary_mbox?
Either Y or N. If Y is specified, then mail from *NJE_userid* `at` *NJE_nodeid* is converted to *nickname*`@smtp-gateway` before it is sent to the TCP recipient. Each *NJE_userid*, *NJE_nodeid* pair can only have one *primary_mbox?* record.

SMTP security data set examples: The following example shows an SMTP security data set:

```
*  Records for Jane Doe, within IBM
JDOE   ALMADEN
JDOE   AUSTIN
*  Records for John Smith, within IBM
SMITH  RALEIGH   JOHNNY  Y  N
SMITH  YORKTOWN  JOHNNY  N  Y
SMITH  DALLAS    JOHNNY  N  N
SMITH  RALEIGH   JSMITH  Y  Y
```

For example, mail sent from the following NJE network addresses through the SMTP gateway is rewritten to the following TCP network addresses. Assume the host name of the gateway is `SMTP-GATEWAY.IBM.COM`.

```
NJE Address              TCP Address
-------------------------------------------------------------------------

JDOE  at ALMADEN         JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM
JDOE  at AUSTIN          JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM
SMITH at RALEIGH         JSMITH@SMTP-GATEWAY.IBM.COM
SMITH at YORKTOWN        JOHNNY@SMTP-GATEWAY.IBM.COM
SMITH at DALLAS          JOHNNY%DALLAS@SMTP-GATEWAY.IBM.COM
```

Mail sent from the TCP network to the following TCP network addresses is forwarded to the following NJE network addresses. Assume the host name of the gateway is `SMTP-GATEWAY.IBM.COM`.

```
TCP Address                              NJE Address
-------------------------------------------------------------------------

JDOE%ALMADEN@SMTP-GATEWAY.IBM.COM        JDOE  at ALMADEN
JDOE%AUSTIN@SMTP-GATEWAY.IBM.COM         JDOE  at AUSTIN
JSMITH@SMTP-GATEWAY.IBM.COM              SMITH at RALEIGH
JOHNNY@SMTP-GATEWAY.IBM.COM              SMITH at RALEIGH
SMITH%DALLAS@SMTP-GATEWAY.IBM.COM        SMITH at DALLAS
```

Rejected mail examples: SMTP rejects mail to or from an unauthorized NJE user. If the mail is from the TCP network, SMTP rejects the RCPT TO command with the error:

```
550 User is not a registered gateway user
```

If the mail is from the NJE network, SMTP rejects the MAIL FROM command with the error:

```
550 User is not a registered gateway user
```

and includes the *mailfiledsprefix*.SECURITY.MEMO data set as an explanation.

The following example shows a sample *mailfiledsprefix*.SECURITY.MEMO data set:

```
The mail you sent to this SMTP gateway cannot be delivered because
you are not a registered user of this gateway.  Contact your local
administrator for instructions on how to be authorized to use this
SMTP gateway.
```

The following is an example of rejected mail that was sent to an unregistered NJE user:

```
Date: Fri, 5 Jul 91 10:55:59 EST
From: SMTP@MVS1.ACME.COM
To: DANIEL@MVS1
Subject: Undeliverable Mail

MVS1.ACME.COM unable to deliver following mail to recipient(s):
    <MATT@SMTP-GATEWAY.IBM.COM>
MVS1.ACME.COM received negative reply from host:
    SMTP-GATEWAY
550 User 'MATT@SMTP-GATEWAY' is not a registered gateway user

           ** Text of Mail follows **
Date: Fri, 5 Jul 91 10:55:56 EDT
From: <DANIEL@MVS1.ACME.COM>
To:   <MATT@SMTP-GATEWAY.IBM.COM>
Subject:  Lunch
Matt,

Do you have time to meet for lunch next week?  I want to discuss the
shipment of ACME iron birdseed.
Daniel
```

The following is an example of rejected mail that was sent from an unregistered NJE user:

```
Date: Fri, 5 Jul 91 11:35:18 EST
From: <SMTP@SMTP-GATEWAY.IBM.COM>
To: <MATT@SMTP-GATEWAY.IBM.COM>
Subject: Undeliverable Mail
Unable to deliver mail to some/all recipients.
050 MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
550-User 'MATT@SMTP-GATEWAY' is not a registered gateway user.
550-
550-The mail you sent to this SMTP gateway cannot be delivered because
550-you are not a registered user of this gateway.  Contact your local
550-administrator for instructions on how to be authorized to use this
550 SMTP gateway.
           ** Text of Mail follows **
HELO SMTP-GATEWAY.IBM.COM
MAIL FROM:<MATT@SMTP-GATEWAY.IBM.COM>
RCPT TO:<DANIEL@MVS1.ACME.COM>
DATA
Date: Fri, 5 Jul 91 11:34:17 EST
From: <MATT@SMTP-GATEWAY.IBM.COM>
```

```
To:   <DANIEL@MVS1.ACME.COM>
Subject:  Awaiting your message
Daniel,
When are you going to contact me about the iron birdseed and giant
electromagnet that I ordered?  I would like to meet with you soon.
Matt

.
QUIT
```

Step 8: Enable SMTP domain name resolution

The SMTP server's RESOLVERUSAGE statement indicates if domain name resolution is to be used or not. If name resolution is not desired, RESOLVERUSAGE NO should be specified. See "Step 9: Enable sending of non-local messages to other mail servers" on page 687.

If the RESOLVERUSAGE statement is not specified or is specified as RESOLVERUSAGE YES, the SMTP server will resolve domain names. Resolver TCPIP.DATA statements must be configured before you can use domain name resolution for SMTP. For a description of how TCPIP.DATA statements can be specified, see "Understanding resolvers" on page 12.

For more information on the SMTP RESOLVERUSAGE statement and the TCPIP.DATA resolver statements, refer to *z/OS Communications Server: IP Configuration Reference*.

To use a domain name server, configure the TCPIP.DATA data set with the IP address of one or more name servers. If the TCPIP.DATA data set does not point to any name servers, the local site tables are used by SMTP. However, if the SMTP server is configured to use name servers, SMTP does not use the site tables.

To determine which DSN the SMTP server is using, look for message number EZA5231I in the output data set specified by the OUTPUT statement in the *hlq*.SEZAINST(SMTPPROC).

When SMTP uses a domain name server, it asks the domain name server for the MX records for the host to which it is trying to connect. If SMTP does not find MX records for a host, it delivers mail only to the primary host listed in the A records. The MX and A records are coded in the domain name server database.

The basic idea behind MX records is to send the mail as close as possible to the final destination. The destination host may currently be inactive, for example, because it is in another time zone. SMTP needs a synchronous connection to deliver the mail, but due to the different time zones, two systems might never be active at the same time and would never be able to exchange mail.

Using MX records would allow the SMTP server to deliver the mail to an alternate host if the first one is unavailable. SMTP tries to deliver mail to the host with the lowest MX record count. If the host is not currently available, it tries the host with the next lowest count.

For example, if SMTP wants to send mail to USER@BASKET, it checks the name server for MX records and finds the following:

```
MVS20  BASKET  A
       BASKET  MX  0    MVS20
       BASKET  MX  5    MVS18
       BASKET  MX  10   VMQ
```

SMTP delivers the mail to the BASKET with the lowest count on its MX record. If `MVS20` is unable to receive the mail, SMTP then tries to deliver it to `MVS18`. If `MVS18` cannot receive the mail, it tries `VMQ`. If none of the hosts can receive the mail, SMTP stores the mail and queues it for later delivery, at which time the process repeats.

For more information about how to add MX records to your name server, consult RFC 974, “Mail Routing and the Domain System.”

To receive a detailed trace on how SMTP is resolving a particular host name, you can issue the SMSG SMTP TRACE command at the console. You can also add the TRACE RESOLVER statement when configuring the TCPIP.DATA data set, but this will also trace the name resolution for all the other applications using the name server. To prevent the console log from becoming too large, only use the TRACE RESOLVER statement for debugging.

If changes to the domain name server requires you to resolve already queued mail again, use the SMSG SMTP EXPIRE command as described in the *z/OS Communications Server: IP User's Guide and Commands*. You can also query operating statistics, such as mail delivery queues of the SMTP server, by using the SMSG SMTP command. This and other administrative tasks are discussed in more detail in the *z/OS Communications Server: IP User's Guide and Commands*.

Step 9: Enable sending of non-local messages to other mail servers

The SMTP server can be configured to send all your non-local TCP/IP SMTP mail to a specified mail server, or mail relay. You may need to do this if you have installed a FIREWALL.

This is accomplished either by making the following changes to your hlq.SMTP.CONFIG data set:

1. Inhibit SMTP from attempting to resolve non-local hostnames by specifying the following statement in your SMTP.CONFIG data set:

   ```
   RESOLVERUSAGE NO
   ```

2. Update the SMTP.CONFIG file to redirect mail to a specific server using the IPMAILERADDRESS statement:

   ```
   IPMAILERADDRESS ip_address
   ```

   where ip_address is address of the mail server that can perform the hostname resolution.

You can also forward any unknown mail to another mailer on the NJE network using the ″MAILER ... UNKNOWN″ statement. For more information, refer to the *z/OS Communications Server: IP Configuration Reference*. Note you *cannot* specify both an IPMAILADDRESS and the ″MAILER ... UNKNOWN″ statements.

Step 10: Design SMTP exit to inspect and filter unwanted mail (optional)

The SMTP exit facility allows an installation to better control the volume of unwanted mail (spam) that is entering the installation. SMTP makes use of the Dynamic Exit Facility (CSVDYNEX macro) provided by MVS. Refer to *z/OS MVS Programming: Authorized Assembler Services Guide*, for more information. The exit is provided by the customer to implement policies that they deem workable. Based on user-defined (and implemented) criteria, individual mail items may be rejected before they consume other resources. SMTPEXIT is provided as a programming guide to aid in the implementation of the local policies. It can be found in

hlq.SEZAINST. This exit must be REENTRANT and AMODE 31, in an authorized library. In using the SMTP exit a name token (EZBTCPIPSMTPEXIT) needs to be established in SYS1.PARMLIB(PROGxx).

If a user program is enabled, message EZA5549I is generated in the SMTP output data set when the SMTPPROC program is started. This message indicates a user exit is active.

This exit can be replaced dynamically without stopping the SMTPPROC program. The procedure for doing this follows:

1. Issue a ″SMSG *smtpprocname* STOPEXIT″ TSO command. The TSO user ID must be in the authorized list for SMTPPROC to issue this command. This will cause SMTP to issue the termination call to the exit and then set a flag so that the exit will not be called anymore. Processing of mail will continue as if there is no exit.
2. Remove the exit via the SETPROG EXIT operator command or by updating SYS1.PARMLIB(PROGxx) and issuing the refresh console command. Example of updating SYS1.PARMLIB follows:
   a. Include the following in SYS1.PARMLIB(PROGxx):
      ```
      EXIT DELETE EXITNAME(EZBTCPIPSMTPEXIT) MODNAME(MYEXIT) FORCE(YES)
      ```
   b. At the MVS console issue SET PROG=xx.
3. Replace with the desired new exit by adding the exit via the SETPROG EXIT operator command or by updating SYS1.PARMLIB(PROGxx). Example of updating SYS1.PARMLIB follows:
   a. In SYS1.PARMLIB(PROGxx) have this line:
      ```
      EXIT ADD EXITNAME(EZBTCPIPSMTPEXIT) MODNAME(NEWEXIT)
      ```
   b. At the MVS console issue SET PROG=xx.
4. Issue a ″SMSG *smtpprocname* STARTEXIT″ TSO command. This will cause SMTP to issue the initialization call to the exit. A flag is then set so the exit will be called from then on for new mail connections. Processing of new mail will continue with the exit being called. The first smtp command to be seen by a reinstated exit will be HELO. The exit will not be called in the middle of a currently processing exchange.

In designing the SMTP exit some of the following design points need to be considered. It should be noted that a remote SMTP application will be connected to the local SMTP while this exit is running. If too much time is spent in the exit, timeout situations may occur and the remote SMTP application may terminate the connection and then go into retry logic. This will seriously affect the performance of the mail system. The exit must be coded as efficiently as possible and all efforts should be taken to avoid excessive processing or waiting, e.g. I/O operations and DNS resolver calls, while within the exit. Efforts to reject mail may be more efficient if extensive scanning of the data portion of the message can be avoided. The exit may allow processing to continue or reject the entire message and does not have the ability to reject individual segments of a message. The message contents cannot be changed in any way by the exit. The exit may accept a message at any point and disable further exit calls for that message. Only commands that are currently implemented by the SMTP program will be passed to the exit program. RFC 2505 and RFC 2635 should be read and understood before undertaking such a coding effort. Multiple connections can occur simultaneously and the exit must take precautions to keep any desired state information on a connection basis. More information on SMTP commands and standards are documented in RFCs 821 and 822.

Refer to the *z/OS Communications Server: IP Configuration Reference* for more detailed information.

Configuring z/OS UNIX sendmail and popper

The following is intended to provide the administrator with specific information on how to configure sendmail on the z/OS platform. Before using this chapter, become familiar with the industry-accepted publication for sendmail, *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-222-0). That publication is known throughout the industry as simply the *bat book*, and this chapter consistently refers to the *bat book* for further information.

Additional information about sendmail can also be found on the z/OS UNIX application Web site, http://www.s390.ibm.com/unix, as well as in documents from the sample directory that were received during the port of sendmail 8.8.7 from the http://www.sendmail.org Web site. The Sendmail Installation and Operation Guide document (sendmail.ps), for instance, is the generic guide from http://www.sendmail.org, which might be helpful as a more thorough guide in a slightly different format. The README.m4 document gives more details for building a configuration file using the m4 preprocessor.

This chapter also provides information on how to configure popper on the z/OS platform. The popper function requires very little configuration. For more information on the protocol used by this UNIX application, see RFC 1939.

Overview

The simple mail architecture in which sendmail and popper fit includes a mail user agent (MUA), a mail transfer agent (MTA), and a mail delivery agent (MDA). An MUA is client software that a user invokes directly to send and receive e-mail. Examples of MUAs include Eudora, Netscape Navigator, pine and elm. An MTA is software that actually routes messages from a sender's system to the receiver's system. sendmail is an MTA. It is worth noting, however, that sendmail relies on other programs to implement non-SMTP based transport (for example, UUCP-based transport as well as local delivery to a user's mail spool file). An MDA is server software that delivers received mail to a user's MUA. Popper is an example of an MDA using the POP3 protocol.

At the sender's end of the mail delivery process, the sender's MUA transmits the message to be delivered to sendmail, as shown in Figure 73.

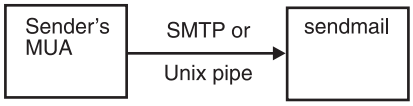


Figure 73. Sender MUA transmits the message to sendmail

This can occur in one of two ways. If the MUA is running on the local host, the message can be transmitted by executing a copy of sendmail and transmitting the message to the standard input of that process via a UNIX pipe.

Alternatively (and more commonly), a copy of sendmail will be running as a daemon, and the MUA (running on either the local host, or on a remote host) will open an SMTP connection to the sendmail daemon, transmitting the message to be delivered via that SMTP connection. In this case, sendmail is acting as an SMTP server, while the MUA is acting as an SMTP client.

In the next step, for each recipient address, sendmail transmits the message to some other SMTP server, to route the message to its final destination at the recipient's site. This is shown in Figure 74.

Figure 74. sendmail transmits the message to an intermediate SMTP server

The receiving SMTP server, in this case, might be a local hub that handles all mail at the sender's site, a remote hub handling all mail at the recipient's site, or an SMTP server at the recipient's host system.

In the next step, sendmail acts as an SMTP client, initiating an SMTP connection with some SMTP server, and then transmitting the message to be delivered to that server, via the SMTP connection.

At the receiver's end of the mail delivery process, a sendmail daemon receives the message from some SMTP client, as shown in Figure 75.

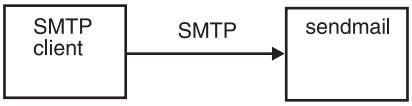


Figure 75. A sendmail daemon receives the message from an SMTP client

The sendmail daemon, acting as an SMTP server, accepts an incoming SMTP connection, and receives a message to be delivered over that SMTP connection. (This is identical to receipt of a message from an MUA, over an SMTP connection.)

Upon receiving the message, sendmail delivers it to the local recipient by appending the message to the recipient's mail spool file. To do this, sendmail requires a local mailer program, as depicted in Figure 76.

Figure 76. sendmail delivers the message to the local recipient

In this step, sendmail executes a specified local mailer program, such as /bin/mail, and transmits the message to be delivered to that mailer via a Unix pipe. The mailer program appends the message to the recipient's mail spool file. With this sendmail's role in delivery of mail is completed.

For the recipient to now read the received message, an MUA must be used. As mentioned in the previous section, depending upon the MUA, this may or may not require an additional MDA, such as popper. If the receiver's MUA has direct access to the mail spool file, the MUA may retrieve the mail directly from the spool file, as depicted in Figure 77 on page 691.

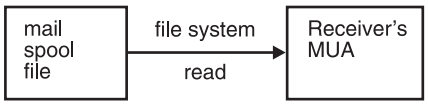


Figure 77. Receiver's MUA has direct access to the mail spool file

Alternatively (and more commonly), the MUA will establish a POP3 connection with a popper daemon, and retrieve the message over that connection. This is shown in Figure 78.

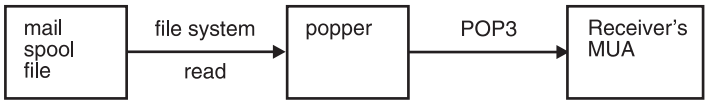


Figure 78. Receiver's MUA retrieves the message over a POP3 connection with a popper daemon

The popper daemon will also allow the receiver's MUA to manage the mail spool file, by allowing it to specify whether and which message should be deleted.

Configuring z/OS UNIX sendmail

This section contains information on the following:

- The sendmail samples directory
- Creating the configuration file
- Creating an aliases file
- Configuration hints and tips

The sendmail samples directory

Much of the sendmail samples directory is dedicated to the automated creation of the configuration file. The /usr/lpp/tcpip/samples/sendmail/cf directory contains a sample.mc file and the subsequent sample.cf configuration file that was created by running the m4 macro preprocessor on the sample.mc file. If the /usr/lpp/tcpip/samples/sendmail directory is examined, the following directory structure can be found:

```
cd /usr/lpp/tcpip/samples/sendmail
ls
README.m4    feature   mailer      siteconfig
cf           hack      ostype      sendmail.ps
domain       m4        sh
```

cf

Both site-dependent and site-independent descriptions of hosts. Files ending in **.mc** (Master Configuration) are the input descriptions. The output is in the corresponding **.cf** file. The general structure of these files is described below.

domain

Site-dependent subdomain descriptions. These are tied to the way your organization wants to do addressing. These descriptions are referenced using the DOMAIN *m4* macro in the **.mc** file.

feature

Definitions of specific features that some particular host in your site might want. These are referenced using the FEATURE *m4* macro. An example feature is `use_cw_file`, which tells z/OS UNIX sendmail to read an `/etc/sendmail.cw` file on startup to find the set of local names.

hack
Local hacks, referenced using the HACK *m4* macro. Avoid these.

m4
Site-independent *m4(1)* include files that have information common to all configuration files. Think of this as a ″#include directory.

mailer
Definitions of mailers, referenced using the MAILER *m4* macro. The mailer types that are known in this distribution are fax, local, smtp, uucp, and usenet. For example, to include support for the UUCP-based mailers, use MAILER(uucp).

ostype
Definitions describing various operating system environments (such as the location of support files). These are referenced using the OSTYPE *m4* macro. This directory contains only the os390 definition.

README
Contains all the latest information regarding this latest version of sendmail from the *www.sendmail.org* site.

sh
Shell files used by the *m4* build process.

siteconfig
Local UUCP connectivity information. These normally contain lists of site information, for example:

- SITE(contessa)
- SITE(hoptoad)
- SITE(nkainc)
- SITE(well)

These are referenced using the SITECONFIG macro:

```
SITECONFIG(site.config.file, name_of_site,X)
```

where *X* is the macro or class name to use. It can be U (indicating locally connected hosts) or one of W, X, or Y for up to three remote UUCP hubs. This directory has been supplanted by the mailer table feature. Any new configurations should use that feature to do UUCP (and other) routing.

sendmail.ps
This is a postscript file of the *Sendmail Installation and Operation Guide* provided by *www.sendmail.org* in this version of sendmail.

Creating the configuration file

The basic steps to create the configuration file are:

1. Retrieve the m4 preprocessor.
2. Create the *.mc* file.
3. Build the configuration file.

Retrieving the m4 preprocessor: Retrieve the m4 macro preprocessor from the z/OS Toys and Tools Web page at http://www.s390.ibm.com/unix/bpxa1toy.html.

The m4 macro preprocessor can be given input that will generate a z/OS UNIX sendmail configuration file. It takes as input a user-defined master configuration

source file (.mc file) that can define mail delivery mechanisms using files provided in the samples directory. For more information on the .mc file, see “Creating the .mc file”.

The m4 preprocessor is downloaded as m4_pax.Z. To *unpax* the file, issue the following command:

```
pax -rzf m4_pax.Z
```

Creating the .mc file: The process of building a z/OS UNIX sendmail configuration file begins by creating a file of m4 statements. The suffix for this file is *.mc*.

The minimal mc file: Every **.mc** file must contain minimal information. This file defines the mail delivery mechanisms understood at this site, how to access them, how to forward e-mail to remote mail systems, and a number of tuning parameters. The following table shows which items are required and also which items are recommended. It is recommended that the starting point for these items be as shown in the sample.mc file, and an investigation of all the m4 techniques that are available to customize the .mc file for your mail server is encouraged [refer to *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-222-0); this book is also referred to as the *bat book*].

Table 24. Required and recommended m4 items

Item	Bat book reference	Required or recommended	Description
OSTYPE()	19.3.1	Required	Support for your operating system
MAILER()	19.3.2	Required	Necessary delivery agent
DOMAIN()	19.3.3	Recommended	Common domain wide information
FEATURE()	19.3.4	Recommended	Solutions to special needs

Example files can be found in the `/usr/lpp/tcpip/samples/sendmail` directory. The *cf* directory contains an example of an **.mc** file. Of special interest are the files that begin with *generic*. These can serve as template statements in developing customized **.mc** files. The following is an example of a simple **.mc** file.

```
divert (-1)
divert(0) dnl
VERSIONID(`OS/390 sample configuration 12/4/97')
OSTYPE(os390)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

Following is a description of these common *m4* items. For more information on these items, refer to the *bat book.*

divert

- (-1) Ignore the lines following.
- (0) Stop diverting and output immediately.

VERSIONID

Used to insert an identifier into each **.mc** and **.m4** file that will become your header.

OSTYPE()

- Support for operating system (the only ostype provided in the `/usr/lpp/tcpip/samples/sendmail/ostype` directory is os390.m4).

- Required.

MAILER()
- Necessary delivery agent.
- Required.
- Known values include:
  - fax
  - local
  - smtp
  - uucp
  - usenet

DOMAIN()
Common domain wide information.

FEATURE()
Solution to special needs.

Building the configuration file: To build the configuration file, go to the directory containing the m4 executable and issue the following command:

```
m4  ../m4/cf.m4 yourmcfile.mc > yourcffile.cf
```

where *yourmcfile* is the name of your .mc file and *yourcffile* is the name you want to give your .cf file.

The `../m4/cf.m4` specifies the master prototype configuration file `cf.m4` in the `m4` directory of the samples/sendmail directory. This is the path to the samples/sendmail directory structure from the location of your m4 executable. This can also be specified in your .mc file using *include* as follows:

```
include('../m4/cf.m4')
```

Creating the aliases file

Aliasing is the process of converting one recipient name into another; a generic name (such as root) into a real user name, or one name into a list of names (that is, a mailing list). Define the location of your aliases file using the AliasFile option in your sendmail.cf file. For example:

```
AliasFile=/etc/aliases
```

For sendmail to work, aliases are required for MAILER-DAEMON and postmaster. Every aliases file must include these required aliases.

The alias for postmaster must expand to the name of a real user, based on the requirement that every site has to be able to accept mail addressed to a user named postmaster. Unless a site has real user account named postmaster, an alias is required in the aliases file. The postmaster receives mail about mail problems sent by mail-related programs and by users that are having trouble sending mail.

When mail is bounced (returned because it could not be delivered), it is sent from MAILER-DAEMON but it is shown as being the original sender who sent the mail. This alias is defined because users often inadvertently reply to the bounced mail.

Following is an example of an aliases file. Lines that begin with # are comments. Empty lines are ignored. For more information on the different forms of aliases, refer to the *bat book.*

```
# Alias for mailer daemon
MAILER-DAEMON:IBMUSER

# Following alias is required by the new mail protocol, RFC 822
postmaster:IBMUSER

# Alias to handle mail to msgs and news
nobody: /dev/null
```

Note: After the aliases file is created and before the sendmail daemon is brought up for the first time, the aliases file must be loaded by running sendmail using the *newaliases* command or with the *-bi* command-line switch.

For more information on the aliases file, refer to the *bat book.*

Configuration hints and tips

This section contains other required or useful information for configuring sendmail. For further information on these topics, refer to the *bat book.*

- SuperUser status is needed to start the sendmail daemon.
- The QueueDirectory option defined in the config file tells sendmail where to queue messages that are temporarily undeliverable. This directory must exist before sendmail is started.
- Sendmail is highly dependent on the Domain Name Server (DNS); it is important that the resolver be set up correctly to avoid unnecessary searching for a user. For more information on DNS, see Chapter 10, “Domain Name System (DNS)” on page 417.
- Table 25 shows the expected file permissions of files that sendmail might use.

Table 25. Sendmail permission table

Path	Type	Owner	Mode		Bat book reference
/	Directory	*root*	0755	drwxr-xr-x	22.1
/usr	Directory	*root*	0755	drwxr-xr-x	18.8.34
/usr/sbin/sendmail	File	*root*	06511	-r-s--s--x	Entire Book
/etc	Directory	*root*	0755	drwxr-xr-x	18.8.34
/etc/sendmail.cf	File	*root*	0644 or 0640		Chapter 27
/etc/sendmail.st	File	*root*	0644	-rw-r--r--	26.6
/etc/sendmail.hf	File	*root*	0444	-r--r--r--	34.8.28
/etc/aliases	File	*root*	0644	-rw-r--r--	Chapter 24
/etc/aliases.pag	File	*root*	0644	-rw-r--r--	24.5
/etc/aliases.dir	File	*root*	0644	-rw-r--r--	24.5
/etc/aliases.db	File	*root*	0644	-rw-r--r--	24.5

If a system has thousands of users defined in the Users list, the administrator might consider enabling the UNIXMAP class. This increases the speed of the security checks performed by sendmail. APAR OW30858 provides details about what is needed to enable the UNIXMAP class.

For additional information about enabling the UNIX map class, refer to *z/OS Security Server RACF Migration*.

sendmail as a daemon

Just as sendmail can transport a mail message over a TCP/IP-based network, it can also receive mail that is sent to it over the network. To do this, it must be run in daemon mode. A daemon is a program that runs in the background independent of terminal control.

As a daemon, sendmail is run once, usually when your machine is booted. Whenever an e–mail message is sent to your machine, the sending machine talks to the sendmail daemon that is listening on your machine.

The -bd command-line switch tells sendmail to run in daemon mode. The -q1h command-line switch tells sendmail to wake up once per hour and process the queue. Command-line switches are described in *z/OS Communications Server: IP User's Guide and Commands*.

Configuring popper

Popper must be invoked by INETD upon the initiation of a TCP connection to the POP3 port 110 (or any other specifically-configured port defined in /etc/services - port 110 is the well-known port for POP3 protocols). Therefore, you must add the following command lines to your /etc/inetd.conf file:

```
pop3 stream tcp nowait bpxroot /usr/sbin/popper popper -d
```

The above must be added to your /etc/inetd.conf file and INETD must be started with this configuration file. For more information on inetd, see *z/OS Communications Server: IP Configuration Reference*.

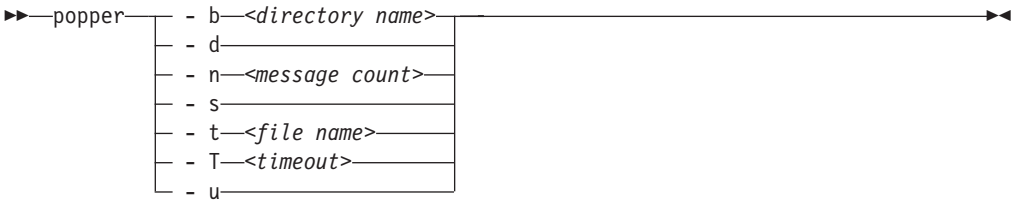
POP3 resides on port 110. You can define additional ports if there is a need for additional command-line options for popper. For information on the options that might be suitable for your site, see the *z/OS Communications Server: IP User's Guide and Commands*.

z/OS UNIX popper will most likely be used by those whose local mailer requires a POP3 server. Typically their administrator will provide them with the address or name of the z/OS running the POP3 server, with instructions on where this information should be used.

Popper command—administering received mail

If the receiver's MUA does not have direct access to the mail spool file, use Popper to access the mail spool on the local host. z/OS Popper will be used when a POP3 server is needed.

Syntax:



Parameters: The following command line options can be used when invoking Popper.

-b *<directory name>*
Specifies the name of the directory in which bulletins are found. If not specified, /usr/mail/bulletins is used as the default.

-d Requests additional debugging messages be turned on.

-n *<message count>*
Specifies the number of old bulletins to be delivered to new users. If not specified, no bulletins are delivered.

-s Requests statistics logging be turned on.

-t *<file name>*
Specifies a trace file for all message logging. If not specified, messages are logged via the syslog facility.

-T *<timeout>*
Specifies the time, in seconds, before an idle POP3 connection is terminated. RFC 1939 specifies a minimum timeout of 600 seconds, but in practice such a long timeout does not work well. (When a connection gets aborted, the user is locked out of his mailbox for the timeout period.) If not specified, 120 seconds is used as the default timeout period.

-u Requests the user's mailbox be updated on abort. RFC 1939 specifies that mailboxes should not be updated (that is, no messages should be deleted) if a connection is aborted abnormally. This option forces an update to occur despite the aborted connection. If not specified, no update will occur on aborted connections.

Chapter 18. TIMED daemon

TIMED is a TCP/IP daemon that is used to provide the time. TIMED gives the time in seconds since midnight January 1, 1900. You can start TIMED from the z/OS shell or as a started procedure. TCP/IP must be started prior to starting TIMED.

Starting TIMED from z/OS shell

TIMED is installed in the /usr/lpp/tcpip/sbin/ directory.

To start the TIMED server from the command line, type the timed command.

```
timed [-l] [-p port]
```

Following are the parameters used for the timed command:

-l Logs all the incoming requests and responses to the system log. Logged information includes the IP address of the requestor.

-p port
The TIMED server usually receives requests on well-known port 37. TIMED uses UDP only. You can specify the port in which requests are to be received.

Starting TIMED as a procedure

The following sample shows how to start TIMED as a procedure.

```
//TIMED    PROC
//*
//*   5694-A01 (C) Copyright IBM Corp. 1997, 2002
//*   Licensed Materials - Property of IBM
//*   This product contains "Restricted Materials of IBM"
//*   All rights reserved.
//*   US Government Users Restricted Rights -
//*   Use, duplication or disclosure restricted by
//*   GSA ADP Schedule Contract with IBM Corp.
//*   See IBM Copyright Instructions.
//*
//* Function: Time server start procedure
//* SMP/E distribution name: EZATTMDP
//*
//TIMED    EXEC PGM=TIMED,REGION=0K,TIME=NOLIMIT,
//         PARM='POSIX(ON),ALL31(ON),TRAP(OFF)/'
//*STEPLIB DD DISP=SHR,DSN=TCP.SEZALOAD,
//*           VOL=SER=,UNIT=
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//         PEND
```

Chapter 19. SNTPD daemon

SNTPD is a TCP/IP daemon that is used to synchronize time between a client and a server. Simple Network Time Protocol (SNTP) is a protocol for synchronizing clocks across a WAN or LAN through a specific formatted message. An External Time Reference (ETR), named stratum 0, is chosen as the highest timer reference used for synchronization. A stratum 1 server is attached to and receives the time from the stratum 0 timer. For example, the z/OS sysplex timer could be a stratum 0 timer, and z/OS Communications Server would be a stratum 1 server. A client attached to the stratum 1 server can also be a stratum 2 server, receiving the time from the stratum 1 server, and so on. SNTP uses UDP packets for data transfer with the well-known port number 123. RFC 2030 (Mills 1996) describes SNTP. You can start SNTPD from the z/OS shell or as a started procedure. Each of these methods is described below. TCP/IP must be started prior to starting SNTPD.

Steps for starting SNTPD from the z/OS shell

Before you begin: Ensure the existence of the following files. The HFS files used by z/OS UNIX SNTPD and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file.

/usr/lpp/tcpip/sbin/sntpd
This is a symbolic link to /usr/lpp/tcpip/sbin/sntpd, which is a sticky-bit file. The SNTPD member of *hlq*.SEZALOAD contains the executable code for the SNTP server.

/usr/lib/nls/msg/C/sntpdmsg.cat
The message catalog used by the z/OS UNIX SNTPD server.

When restricting low port usage, the port used by SNTPD (default value of 123) should either:

- Be reserved for the name of the SNTPD start procedure
- Use the SAF parameter on the PORT statement to restict access to the SNTPD port

Note: There is no configuration file specifically for SNTPD.

Perform the following step to start SNTPD from the z/OS shell:

1. Type sntpd & on the command line. This will start sntpd and send it to the background.

   Following are the optional parameters used for the sntpd command:

   -d Enable debugging and activity logging. Activity logging and debugging messages are written to stdout.

   -df *HFS-pathname*
   Enable debugging and activity logging, and write debugging and activity logging messages to the specified HFS file. For example:

   ```
   -df /var/sntpd.debug
   ```

   -pf *HFS-pathname*
   HFS path for pid file. For example:

```
-pf /var
```

-m *nnnnn*
: Act in multicast mode. Send multicast updates (TTL=1) on all interfaces every *nnnnn* seconds. Listen for requests and respond with unicast replies. The valid range for *nnnnn* is 1 to 16284.

-b *nnnnn*
: Act in broadcast mode. Send local broadcasts on all interfaces every *nnnnn* seconds. Listen for requests and respond with unicast replies. The valid range for *nnnnn* is 1 to 16284.

-?
: Display the syntax of the command usage and options.

You know that SNTPD has started when the following message appears in the z/OS UNIX shell:

```
EZZ9600I SNTP server is ready.
```

Steps for starting SNTPD as a procedure

Before you begin: Obtain a copy of this sample procedure from SEZAINST and store it in one of your PROCLIB concatenation data sets.

Perform the following step to start SNTPD as a procedure:

1. Invoke the procedure using the system operator start command. The following sample [shipped as *hlq*.SEZAINST(SNTPD)] shows how to start SNTPD as a procedure:

```
//SNTPD PROC
   //*
   //* Communications Server IP
   //* SMP/E DISTRIBUTION NAME: EZASNPRO
   //*
   //* 5694-A01 (C) COPYRIGHT IBM CORP. 2002.
   //* LICENSED MATERIALS - PROPERTY OF IBM
   //* THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM"
   //* ALL RIGHTS RESERVED.
   //* US GOVERNMENT USERS RESTRICTED RIGHTS -
   //* USE, DUPLICATION OR DISCLOSURE RESTRICTED BY
   //* GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
   //* SEE IBM COPYRIGHT INSTRUCTIONS.
   //*
   //* FUNCTION: SNTP DAEMON START PROCEDURE
   //*
   //SNTPD EXEC PGM=SNTPD,REGION=4096K,TIME=NOLIMIT,
   // PARM='POSIX(ON),ALL31(ON),TRAP(OFF)/ -d'
   //SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
   //SYSIN DD DUMMY
   //SYSERR DD SYSOUT=*
   //SYSOUT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
   //CEEDUMP DD SYSOUT=*
   //SYSABEND DD SYSOUT=*
   //*
```

You know that SNTPD has started when the following message appears on the console:

```
EZZ9600I SNTP server is ready.
```

Stack affinity

If you are running in a multiple stack environment and want SNTPD to use only a single stack, the environment variable _BPXK_SETIBMOPT_TRANSPORT can be used.

Chapter 20. Remote Execution

This chapter describes how to configure and operate both the Remote Execution server and the UNIX Remote Execution server. z/OS Communications Server supports remote execution daemons in both the UNIX and TSO environments. To execute commands under the UNIX shell, use the UNIX REXEC or UNIX RSH server. To execute commands under TSO, use the TSO REXEC and RSH servers. The main difference between using REXEC versus RSH is that RSH gives you the option of allowing the command execution without having to specify a password.

UNIX REXEC

The UNIX Remote Execution Protocol Daemon (REXECD) is the server for the REXEC routine. REXECD allows execution of z/OS UNIX commands with authentication based on user names and passwords.

The Remote Shell Server (RSHD) is the server for the remote shell (RSH) client. The server provides remote execution facilities with authentication based on privileged port numbers, user IDs, and passwords.

See “Configuring the z/OS UNIX Remote Execution servers” on page 708 for more information about configuring this server.

TSO REXEC

The TSO Remote Execution server allows execution of a TSO command that has been received at a remote host. This server runs the Remote EXEcution Command Daemon (REXECD) which supports both the Remote Execution (REXEC) and Remote Shell (RSH) protocols.

This chapter describes how to configure and operate the Remote Execution server.

Configuring the TSO Remote Execution server

Steps to configure the TSO Remote Execution server:

1. Update the Remote Execution cataloged procedure.
2. Update AUTOLOG and PORT statements in the PROFILE.TCPIP data set.
3. Determine whether the Remote Execution client will send a Remote Execution (REXEC) command or Remote Shell (RSH) command.
4. Permit remote users to access MVS resources. (Required only if the client is not sending a password.)
5. Create a user exit routine (optional).

Step 1: Configuring PROFILE.TCPIP for TSO Remote Execution server

If you want the Remote Execution server to start automatically when the TCPIP address space is started, include the name of the member containing the RXSERVE cataloged procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP data set.

```
AUTOLOG
  RXSERVE
ENDAUTOLOG
```

To ensure that port 512 is reserved for the Remote Execution protocol and port 514 for the Remote Shell protocol, add the name of the member containing the Remote Execution cataloged procedure to the PORT statement in *hlq*.PROFILE.TCPIP:

```
PORT
  512 TCP RXSERVE
  514 TCP RXSERVE
```

Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the AUTOLOG and PORT statements.

Step 2: Determine whether Remote Execution client will send REXEC or RSH commands

The Remote Execution client can send commands to the TSO Remote Execution server by the following methods:

1. Sending the Remote Execution (REXEC) command
2. Sending the Remote Shell (RSH) command with a user ID and password separated by a slash (/) character with the -l option on the RSH command
3. Sending the Remote Shell (RSH) command without a password

With methods 1 and 2, the TSO Remote Execution server executes the request and passes the password to MVS for verification. (REXEC commands require a password.) When these methods are used, skip Step 3.

With method 3, to enable an RSH client to send RSH commands to the TSO Remote Execution server without specifying a password, Step 3 is required.

Step 3: Permit remote users to access MVS resources (optional)

This step is necessary only if your installation allows users to issue remote execution commands without the requirement of specifying a password on the remote execution client.

Use the following steps to ensure that the server can correctly access necessary MVS resources. You can use z/OS Security Server (RACF) or an equivalent security program.

1. Verify that your system has been configured for allowing surrogate job submission as described in *z/OS Security Server RACF Security Administrator's Guide* (SC28-1915) or by using an equivalent security program.
2. Authorize the TSO Remote Execution server to submit jobs for the MVS user ID specified with the -l option of the RSH command. This can be done with the RACF facility as described in *z/OS Security Server RACF General User's Guide* (SC28-1917), or by using an equivalent security program.
3. Define an *mvs_userid*.RHOSTS.DATA data set and authorize the TSO Remote Execution server userid permission to read this data set. This can be done with the RACF facility as described in *z/OS Security Server RACF General User's Guide* (SC28-1917), or by using an equivalent security program.

   Note: This is the userid used to start the RXSERVE address space.

   This data set identifies the Remote Execution clients that can execute MVS commands remotely by sending an RSH command.

   When a Remote Execution client sends an RSH request to the TSO Remote Execution server, the request includes the local user ID of the client user

(*local_userID*) and, if the client user specified the -l option of the RSH command, the request also contains the user ID to use on the remote host (*mvs_userid*). If the client does not specify the -l option, the user ID to be used on the remote host is assumed to be the same as the *local_userID*.

When the TSO Remote Execution server receives an RSH command without a password, the server looks for a data set called *mvs_userid*.RHOSTS.DATA. The *mvs_userid*.RHOST.DATA data set contains one or more entries. Each entry consists of two parts, a fully qualified name of the client user's host and a *local_userID* associated with that host. The *local_userID* is case sensitive. If the data set exists, the server reads it and looks for an entry with a host name that matches the client user's host. If the user ID specified on this entry in the RHOSTS.DATA data set matches the *local_userID* passed on the RSH command, the RSH command continues processing. If the entry does not exist, the server responds to the client with message `EZA4386E Permission denied`.

In the following example of an RHOSTS.DATA data set, the MVS client user `mvsuser` is allowed to issue the RSH command without a password from host `rs60007` with a local AIX user ID of `mvsuser`.

Example of mvsuser.RHOSTS.DATA data set:

```
rs60007.itso.ral.ibm.com mvsuser
```

4. Users may be authenticated using Kerberos or GSS. If the username in the Kerberos or GSS credentials matches the local user ID (*local_userID*) of the client supplied by the RSH client, then no password is required.

Step 4: Update the TSO Remote Execution cataloged procedure

Update the TSO Remote Execution cataloged procedure by copying the sample provided in *hlq*.SEZAINST(RXPROC) to your system or recognized PROCLIB and modifying it to suit your local conditions. Specify the TSO Remote Execution server parameters and modify the JCL as required for your installation.

You can update the TSO Remote Execution server operating parameters during execution with the MODIFY command. All but MAXCONN can be changed.

Step 5: Create a user exit routine (optional)

Optionally, you can provide a user exit routine. This routine can be used to alter the JOB and EXEC statement parameters to meet installation-specific requirements such as which system should process the job and/or environment unique accounting information prior to submission of the TSO batch job.

The user exit should have the AMODE(31) and RMODE(24) attributes to provide addressability to the input parameters.

On entry to the user exit, register 1 points to the following parameter list:

Offset	Description
0	A pointer to a mixed INET address
4	A pointer to JOB statement parameters
8	A pointer to EXEC statement parameters
12	A pointer to an optional JES control statement

The INET address consists of the following fields:

Offset	Description
0	2 bytes (AF_INET or 2)

2 2 bytes (server port)
4 4 bytes (client INET address)

The JOB statement parameters can be up to 1024 characters in length and are ended by X'00'. You can modify the parameters with the exit routine. Upon entry, the parameters are set to:

- *user_ID*
- USER=*user_id*
- PASSWORD=*password*
- MSGCLASS=*msgclass*

MSGCLASS is as specified in the Remote Execution cataloged procedure. *userid* and PASSWORD are as received from the requesting client.

For RSH commands without passwords, note that the PASSWORD= parameter is not present. The *userid* in the first positional parameter can be processed by an installation-written JES exit.

The EXEC statement parameters can be up to 256 bytes in length and are ended by X'00'. These parameters can be modified by the exit routine. On entry, it contains the EXEC statement for the procedure specified in the TSOPROC parameter of the Remote Execution server or the default IKJACCNT procedure if TSOPROC is not specified.

The JES control statement parameter can be up to 256 bytes in length and is ended by X'00'. Upon entry, the parameter field is set to X'00'. Any JES control statement added by the user exit will be put between the JOB and the EXEC statement.

The modified JOB and EXEC statements are submitted as a TSO batch job.

The user exit is shipped as a sample in the RXUEXIT member of the SEZAINST data set. Refer to the REXEC chapter in *z/OS Communications Server: IP Configuration Reference* for more information about this sample.

Configuring the z/OS UNIX Remote Execution servers

Installation information

This section describes the HFS files used by z/OS UNIX REXECD and RSHD.

HFS files for z/OS UNIX REXECD

Note: The userid associated with the daemon in /etc/inetd.conf requires superuser authority. Refer to *z/OS UNIX System Services Planning* for a description of the kinds of authority defined for daemons.

The HFS files used by z/OS UNIX REXECD and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/orexecd
The server.

If BPX.DAEMON is specified, then the sticky bit must be set on, and /usr/sbin/orexecd, and orexecd can reside in an authorized MVS data set.

/usr/lib/nls/msg/C/rexdmsg.cat
The message catalog used by the z/OS UNIX REXECD server.

Note: This is not an actual member at this location, but it is a symbolic link to the part in /usr/lpp/tcpip/nls/msg/C/*.

Where the server looks for the message catalog (rexdmsg.cat) depends on the value of NLSPATH and LANG environment variables. If you want to store the msg.cats elsewhere, you need to change the NLSPATH or the LANG environment variables. If rexdmsg.cat does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

HFS files for z/OS UNIX RSHD

The HFS files used by z/OS UNIX RSHD and their locations in the HFS are as follows:

/etc/services
The ports for each application are defined here.

/etc/syslog.conf
The configuration parameters for usage of syslogd are defined in this file.

/etc/inetd.conf
The configuration parameters for all applications started by inetd are defined in this file.

/usr/sbin/orshd
The server.

If BPX.DAEMON is specified, the sticky bit must be set on, and /usr/sbin/orshd, and orshd can reside in an authorized MVS data set.

/usr/sbin/ruserok
An optional user exit that will authenticate users logging into the z/OS UNIX RSHD server with a null password. See “Setting up the z/OS UNIX RSHD installation exit” on page 710 below for more information.

Note: This exit is required to allow support for null passwords with RSH.

/usr/lib/nls/msg/C/rshdmsg.cat
The message catalog associated with the z/OS UNIX RSHD client is stored here. If this file does not exist, the software will default to the messages hard-coded within the software. These messages duplicate the English message catalog that is shipped with the product.

Note: The message catalog is not actually stored here. This is a symbolic link, and the actual member is in /usr/lpp/tcpip/nls/msg/C/*.

Setting up the z/OS UNIX RSHD installation exit

When the -r option is enabled, if there is no password specified on the RSH command from the client, z/OS UNIX RSHD will drive the installation exit. When the installation exit is driven, RSHD looks for a program in /usr/sbin named ruserok. This is the only name that it will look for. If /usr/sbin/ruserok is not found, the request will fail.

When the z/OS UNIX RSHD server invokes /user/sbin/ruserok, it will pass parameters in the following order:

1. Host name or the host IP address
2. Local user's UID
3. Remote userid
4. Local userid

If z/OS UNIX RSHD receives a return code of zero from the installation exit, z/OS UNIX RSHD continues. Any nonzero return code from the installation exit will cause RSHD to issue message EZYRS25E to the client and terminate all connections. The following code fragment can be used as an example to begin building a working ruserok installation exit:

```
int main(argc, argv)
     int argc;
     char *argv[];
  char *rhost1;    /* "hostname" or "hostname.domain" of client
                      obtained by caller:
                      gethostbyaddr(getpeername()) or the host
                      ip address used by the gethostbyaddr if
                      it failed to return a "hostname" */
  int locuid;      /* uid of the user name on local system */
  char *cliuname;  /* user name on client's system */
  char *servuname; /* user name on this (server's) system */
  int rc = 4;

  rhost1 = argv[1];
  locuid = atoi(argv[2]);
  cliuname = argv[3];
  servuname = argv[4];
  .
  <authenticate user and set rc=0 if valid>
  .
  return(rc);
```

Configuring TSO and z/OS UNIX Remote Execution servers to use the same port

Since the remote execution servers are generic servers, they attempt to bind to INADDR_ANY when they are started. This allows them to listen on all defined IP addresses. However, this prevents both the TSO and z/OS UNIX Remote Execution servers from listening on the same port, and one of the servers would have to use a nonstandard port. Using the BIND parameter on the PORT reservation statement in the TCPIP profile data set allows both the TSO and z/OS UNIX Remote Execution servers to bind to the same ports using different IP addresses. The following steps illustrate how this can be done. For more information on the PORT reservation statement, see *z/OS Communications Server: IP Configuration Reference*.

1. Define a VIPA address to the TCPIP profile data set. This example shows a static VIPA address, but either a static or dynamic VIPA can be used.

```
DEVICE VIPAD1 VIRTUAL 0
LINK   VIPA1  VIRTUAL 0 VIPAD1

HOME
134.134.134.36       VIPA1
```

2. Add PORT statements to the TCPIP profile for both the TSO and z/OS UNIX Remote Execution servers. One of the servers will bind to the VIPA address. The other can bind to INADDR_ANY by not specifying the BIND parameter. In this example, the z/OS UNIX Remote Execution servers will bind to the VIPA address. Also update the /etc/services file so that exec uses 512 and shell uses 514.

```
512 TCP OMVS BIND 134.134.134.36 ; z/OS Unix REXECD
514 TCP OMVS BIND 134.134.134.36 ; z/OS Unix RSHD
512 TCP RXSERVE                  ; TSO REXECD
514 TCP RXSERVE                  ; TSO RSHD
```

   It is important that the server with the BIND parameter is listed before the one without the BIND parameter. This setup directs all requests to ports 512 or 514 with a destination IP address of 134.134.134.36 to the z/OS UNIX Remote Execution servers. Requests to ports 512 or 514 with a local destination IP address that is not 134.134.134.36 are directed to the TSO Remote Execution server.

To verify this setup:

1. Start the stack with the TCPIP profile changes described above and start RXSERVE and INETD.

   Note: INETD listens for the REXEC and RSH servers under z/OS UNIX.

2. Issue NETSTAT and it should show that both the REXEC servers are listening on port 512 and both RSH servers are listening on port 514. INETD, which listens for the z/OS UNIX Remote Execution servers, only listens on the VIPA address.

```
MVS TCP/IP NETSTAT CS V1R2       TCPIP NAME: TCPCS            21:34:41
User Id  Conn     Local Socket           Foreign Socket         State
-------  ----     ------------           --------------         -----
INETDCS1 0000000D 134.134.134.36..514    0.0.0.0..0             Listen
INETDCS1 0000000E 134.134.134.36..512    0.0.0.0..0             Listen
RXSERVE  00000019 0.0.0.0..514           0.0.0.0..0             Listen
RXSERVE  00000018 0.0.0.0..512           0.0.0.0..0             Listen
```

Chapter 21. Miscellaneous (MISC) server

The Miscellaneous (MISC) server is a server that can be used to test and debug applications.

The MISC server supports the 3 protocols described in RFCs 862, 863, and 864:
- Discard
- Echo
- Character Generator

Discard protocol

The MISC server simply throws away any data it receives. A TCP-based server listens for TCP connections on TCP port 9. If a connection is established, the data is discarded and no response is sent. A UDP-based server listens for UDP datagrams on UDP port 9. When a datagram is received, it is discarded and no response is sent.

Echo protocol

The MISC server returns to the originating application any data that it receives. A TCP-based server listens for TCP connections on TCP port 7. Once a connection is established, any data that is received is sent back to the originating application. A UDP-based server listens for UDP datagrams on UDP port 7. When a datagram is received, the data it contained is sent back as an answering datagram.

Character generator protocol

The MISC server sends a repetitive stream of character data without regard to its content. A TCP-based server listens for TCP connections on TCP port 19. When a connection is established, a stream of data is sent to the connecting application. Any data that is received is thrown away. A UDP-based server listens for UDP datagrams on port 19. When a datagram is received, an answering datagram is sent that contains a random number (between 0 and 512) of characters. The data in the received datagram is ignored.

The data that is generated follows an ordered sequence. It repeats a pattern of 94 printable ASCII characters in a ring, so that character number 0 follows character number 94.

Following is an example of the repeated pattern.

```
!"#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefgh
"#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghi
#$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghij
$%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijk
%&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijkl
&'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklm
'()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmn
()*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmno
)*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnop
*└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopq
└,-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqr,-
./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrs
-./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrst
./0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstu
/0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuv
0123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvw
123456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwx
23456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxy
3456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz
456789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz{
56789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz{|
6789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz{|}
789:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz{|}~
89:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz{|}~
9:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz{|}~ !
:;◄=►?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]¬_`abcdefghijklmnopqrstuvwxyz{|}~ !"
```

Configuring the MISC server

1. Specify AUTOLOG and PORT statements in *hlq*.PROFILE.TCPIP.
2. Update the MISC server cataloged procedure (MISCSERV).

Step 1: Configuring PROFILE.TCPIP for the MISC server

To allow the MISC server to start automatically when TCPIP is initialized, include the member name of the MISC server cataloged procedure in the AUTOLOG statement in the *hlq*.PROFILE.TCPIP.

```
AUTOLOG
  MISCSERV
ENDAUTOLOG
```

The AUTOLOG entry in *hlq*.PROFILE.TCPIP is optional. You can choose to start the MISC server manually, when it is needed, using the START command:

```
START MISCSERV
```

The MISC server requires ports 7, 9, and 19 for both TCP and UDP. To ensure that these ports are reserved for the MISC server, verify that they are assigned to the member containing the MISC server cataloged procedure in the PORT statement in PROFILE.TCPIP.

```
PORT
    7 UDP MISCSERV
    7 TCP MISCSERV
```

```
 9 UDP MISCSERV
 9 TCP MISCSERV
19 UDP MISCSERV
19 TCP MISCSERV
```

For more information on these statements, see the *z/OS Communications Server: IP Configuration Reference*.

Step 2: Updating the MISC server cataloged procedure (MISCSERV)

Update the MISC server cataloged procedure by copying the sample in *hlq*.SEZAINST(MISCSERV) to your system or recognized PROCLIB and modifying the parameters and data set names to suit your local conditions.

MISC server cataloged procedure (MISCSERV)

```
//MISCSERV PROC MODULE=MISCSRV,PARMS=''
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: SEZAINST(MISCSERV)
//*
//*        Licensed Materials - Program Property of IBM.
//*        "Restricted Materials of IBM"
//*        5694-A01 (C) COPYRIGHT IBM CORP. 1994, 2001
//*        Status = CSV1R2
//*        Distribution library SEZAINST(MISCSERV)
//*
//MISCSERV EXEC PGM=&MODULE,
//           REGION=4096K,TIME=1440,
//           PARM='&PARMS'
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here. If you add
//*        them to STEPLIB, they must be APF authorized.  Change
//*        the name as appropriate for your installation.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//*        MSMISCSR identifies an optional data set for NLS support.
//*        It specifies the MISC server message repository.
//*
//*MSMISCSR DD DSN=TCPIP.SEZAINST(MSMISCSR),DISP=SHR
//*
//*       SYSTCPD explicitly identifies which data set is to be
//*       used to obtain the parameters defined by TCPIP.DATA
//*       when no GLOBALTCPIPDATA statement is configured.
//*       See the IP Configuration Guide for information on
//*       the TCPIP.DATA search order.
//*       The data set can be any sequential data set or a member of
//*       a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Specifying the MISC server parameters

The MISC server generates periodic messages whenever a client sends data to ports 7, 9, or 19. If this server runs continually for a long period of time, considerable amounts of spool space can be consumed. Therefore, the MISC server has all tracing turned off by default.

You can enable the trace options for any of the three MISC server protocols using the PARMS= parameter on the PROC statement of the cataloged procedure. These options will be in effect when the server starts.

TRACE

Turns on tracing for any of the specified protocols and must be followed by one or more of these three keywords:

ECho Specifies tracing for the echo protocol on port 7.

DIscard

Specifies tracing for the discard protocol on port 9.

CHargen

Specifies tracing for the character generator protocol on port 19.

DEbug

Specifies tracing for problem determination .

For example, the following statement turns tracing on for the echo and discard protocols.

```
//MISCSERV PROC MODULE=MISCSRV,PARMS='TRACE ECHO DISCARD'
```

Part 3. Appendixes

Appendix A. Setting up the inetd configuration file

inetd is a generic listener program used by such servers as z/OS UNIX telnet server and z/OS UNIX rexec server. Other servers such as z/OS UNIX ftp server have their own listener program and do not use inetd.

inetd.conf is an example of the user's configuration file. It is stored in the /etc directory. Ensure that the inetd services required on your system are enabled using configuration statements like those in the following example:

```
#=====================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
shell    stream    tcp        nowait  OMVSKERN /usr/sbin/orshd orshd -l
exec     stream    tcp        nowait  OMVSKERN /usr/sbin/orexecd orexecd -LV
otelnet  stream    tcp        nowait  OMVSKERN /usr/sbin/otelnetd otelnetd  -LV
```

Figure 79. Adding applications to /etc/inetd.conf

If the rshd, rexecd, or otelnetd service is to support IPv6 clients, then *tcp6* should be specified instead of *tcp*. Kerberos is not supported for IPv6-enabled services, such as z/OS UNIX Telnet, z/OS UNIX rsh, and z/OS UNIX rexec.

To establish a relationship between the servers defined in the /etc/inetd.conf file and specific port numbers in the z/OS UNIX environment, insure that statements have been added to ETC.SERVICES for each of these servers. See the sample ETC.SERVICES installed in the /usr/lpp/tcpip/samples/services directory for how to specify ETC.SERVICES statements for these servers.

The traces for both the z/OS UNIX rexec and rsh servers are enabled through options in the inetd configuration file (/etc/inetd.conf):

```
#=====================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#=====================================================================
#
shell      stream    tcp       nowait OMVSKERN /usr/sbin/orshd orshd -d  1
exec       stream    tcp       nowait OMVSKERN /usr/sbin/orexecd orexecd -d  2
```

Figure 80. Setting traces in /etc/inetd.conf

The traces are turned on for both servers by passing a -d argument to the server programs. **1** is the RSHD server and **2** is the REXECD server. All commands executed after the debug flags have been turned on in the inetd configuration file and the inetd server has reread the file will produce trace output.

The trace is written in formatted form to the syslogd facility name daemon with a priority of debug. The trace data can be routed to a file in your Hierarchical File System by specifying the following definition in your syslogd configuration file (/etc/syslogd.conf):

```
  #
  # All ftp, rexecd, rshd
  # debug messages (and above
  # priority messages) go
# to server.debug.a
#
daemon.debug              /tmp/syslogd/server.debug.a
```

In this example, the trace data is written to /tmp/syslogd/server.debug.a in your Hierarchical File System. For more information on syslogd, refer to “Logging of system messages” on page 39.

For more information about inetd, refer to *z/OS UNIX System Services Planning* or *z/OS UNIX System Services Command Reference*.

Appendix B. TLS/SSL security

This appendix is a TLS/SSL reference for the z/OS TN3270 Telnet server, the FTP server and the Digital Certificate Access Server (DCAS). The gskkyman utility and RACF are used as examples for certificate and keyring creation and management. References to RACF apply to any other SAF-compliant security products which contain the required support. Host On Demand V4 running on NT is used as a sample Telnet client.

An overview of Secure Socket Layer (SSL) is given first, followed by the detailed steps needed to perform authentication and encryption at the following levels:

- Using gskkyman
  - Server Authentication only
  - Client Authentication Level 1
- Using RACF
  - Server Authentication only
  - Client Authentication Level 1
  - Client Authentication Level 2
  - Client Authentication Level 3

Additional information about the concepts of cryptography and SSL can be found at the following Web sites:

- http://home.netscape.com/eng/ssl3/
- http://www.verisign.com/repository/crptintr.html

Secure Socket Layer overview

SSL provides data privacy and integrity as well as server and client authentication based upon a Public Key Infrastructure (PKI) method. PKI requires that the server organization generate a public key/private key pair that can be used during negotiations. PKI requires that data encrypted with the public key be decrypted by only the private key and that data encrypted with the private key be decrypted by only the public key. This is considered an asymmetric encryption method because different keys are used at each end of the secure connection. The Server sends its public key to the client when the client requests a connection.

The client and server encrypt SSL parameter negotiations using the PKI method of encryption. One of the most important items negotiated is the encryption algorithm to be used during data transmission. The algorithm chosen will be one that uses the same key at each end of the secure connection. This is known as a symmetric encryption method and is about 1000 times faster than the asymmetric PKI method used during SSL parameter negotiation. The encryption key used by the symmetric encryption method is created and exchanged during SSL negotiation protected by the PKI encryption method.

Some client-server connections support negotiations to determine if the client wants or supports SSL prior to beginning the SSL handshake. Most servers and clients can be configured to immediately start the SSL handshake process or to negotiate whether or not to perform the SSL handshake. Refer to the security information in the appropriate server or client chapters for information on whether negotiated TLS/SSL is supported and how it is implemented.

The SSL protocol begins with the handshake. During the handshake:

- Server authentication is done by the client.
- Optional client authentication is done by the server.
- An encryption algorithm and single encryption key are chosen to encrypt and decrypt session data between the client and server.

Server authentication

When using SSL to secure communications, the SSL authentication mechanism known as Server Authentication is used. This is the minimum amount of security provided by SSL and allows the client to validate that the Server is what it says it is.

To ensure that someone has not stolen the server's private and public keys and is pretending to be the server, the server sends additional information with the public key so the client can confirm the identity of the server. The complete package of information sent to the client is called a digital certificate which conforms to the X.509 standard.

This X.509 digital certificate includes, among other things, the Distinguished Name (DN) of the Server organization, the public key created by the server organization, the Distinguished Name of the organization issuing the certificate, and the issuer's signature. The organization issuing the certificate may be a well-known Certificate Authority (CA) or you may issue (create) your own certificate, called a self-signed certificate.

To create a signature, the certificate issuer first generates a message digest from the owner's DN, the owner's public key, and the issuer's DN. The message digest is the result of hashing this information down to a small size (usually 128 or 160 bits). The message digest result is unique for that information based on the hashing algorithm used. The message digest is encrypted with the issuer's private key creating the issuer's signature.

When the client receives the server certificate, the client must have the public key of the certificate signer. The public key is used to decrypt the message digest. The server certificate also contains the hashing algorithm used to create the message digest. The client uses the same algorithm to create another message digest using the Distinguished Names and public key information in the received server certificate. If this new message digest exactly matches the decrypted message digest (issuer's signature) created by the certificate issuer, the client can be assured that the certificate has not been altered. This method of authentication is dependent on the security of the private key that is used by the certificate issuer.

To conduct commercial business on the Internet, you should obtain a server certificate signed by a well-known Certificate Authority. Server certificates issued by a well-known CA gives the client high assurance that the server is authentic. Most client keyrings have been primed with several well-known CA's certificates. That enables the client to authenticate a Server certificate signed by a well-known CA without having to first obtain the issuer's certificate which includes the public key. For relatively small, private networks within your own enterprise you can create your own self-signed server certificate. The only difference between a CA issued certificate and a self-signed certificate is the issuer's Distinguished Name and who's private key was used to encrypt the message digest. The client needs to use the correct public key to decrypt the message digest. The CA certificate containing the CA's public key is probably already in the client's keyring and it can be used to decrypt the CA signature (message digest). The self-signed certificate containing the organization's public key needs to be added to the client's list of signer

certificates so the client can decrypt the signature (message digest) created when the self-signed certificate was created. Some client products allow the client to add the server certificate to its list of signer certificates when the server certificate is received during SSL negotiation. If the client is confident the certificate really came from the correct server, this is an easy way to add the certificate rather than getting a copy and adding it manually.

For server authentication to work, the server must have a private key and associated server certificate in the server key database file. The gskkyman utility or RACF Common Keyring support can be used to manage the keys and certificates needed for SSL support. If the gskkyman utility was used to create the keyring, a password stash file is also required.

SSL requires a server certificate as part of its server authentication process. The server certificate and the Certificate Authority certificates are stored in a keyring (also referred to as a key database). The server's keyring can be created using the gskkyman utility provided by the System Secure Socket Layer (System SSL) element of z/OS or by using RACF's certificate management support. The keyring is associated with a server or client using server or client specific statements.

Note: Global step-up type certificates are not supported by the Telnet TN3270 server if the client application sends the handshake complete message to the server before completing the second handshake.

Client authentication

Client authentication provides additional authentication and access control by checking client certificates at the server. This support prevents a client from obtaining a connection without an installation approved certificate.

The server authenticates the client by receiving the client's certificate during the SSL handshake and verifying the certificate is valid. Validation is done by the server the same way the client validates the server's certificate. The client sends a signed certificate to the server. System SSL at the server decrypts the signature (message digest) using the public key of the client certificate issuer found in the server key database file. The server then creates a new message digest using the certificate's Distinguished Names and public key and compares the new message digest with the decrypted one. If they match, the server can be assured the client is authentic. Depending on where the client certificate is stored, up to three different levels of client authentication are available. Refer to the security information in the appropriate server or client chapters for setup details.

Level 1 authentication is performed by system SSL. The client passes an X.509 certificate to the Server as part of the SSL Handshake. To pass authentication, the Certificate Authority (CA) that signed the client certificate must be considered trusted by the server. That is, the certificate for the CA must be in the keyring used by the Server and designated as trusted. Note that the value of this option alone is based on which CAs are considered trusted. If the CA is a public CA and the certificate is in an easily obtained class, anyone can obtain such a certificate. In this case passing level 1 SSL Client Authentication does not provide much additional security unless coupled with the level 2 RACF support described below. If the CA is controlled by the enterprise, some level of access control is provided because the client that possesses such a certificate is at least known to the organization.

Level 2 authentication requires that the client certificate be registered with RACF (or other SAF compliant security product) and mapped to a user ID. This is in

addition to the checking done with the first level of client authentication support. The client certificate received during the SSL handshake is used to query the security product to verify that the certificate maps to a user ID known to the system prior to connection negotiation. This level of support provides additional access control at the server and ensures that the end user is known to have a valid user ID on the server host. Each server uses the returned user ID in a different way. Refer to the security information in the appropriate server or client chapters to see how the user ID is used for a particular server. Level 1 authentication is performed prior to level 2 authentication.

Level 3 authentication provides, in addition to level 1 and level 2 support, the capability to restrict access to the server based on the user ID returned from RACF. In some cases a certificate may be valid and mapped to a user ID but should be valid for only one of several servers. The third level of control uses the SERVAUTH RACF class to restrict access to the server based on client user ID. If the SERVAUTH class is not active or the SERVAUTH profile for the server is not defined, it is assumed level 3 authentication is not requested. If the SERVAUTH class is active and the server profile is defined, a connection is accepted only if the requestor's user ID associated with the client certificate is in the profile. Otherwise, the connection is dropped. Refer to "Add user IDs to the SERVAUTH profile access list" on page 737 for RACF setup details.

To enable Client Authentication for each server, use the following server-specific statements:

SERVER TYPE	FTP	TN3270	DCAS
STATEMENT	**SECURE_LOGIN**	**CLIENTAUTH**	**CLIENTAUTH**
AUTH LEVEL 1	REQUIRED	SSLCERT	LOCAL 1
AUTH LEVEL 2	VERIFY_USER	SAFCERT	LOCAL 2
AUTH LEVEL 3	VERIFY_USER	SAFCERT	LOCAL 3

Level 1 client authentication is done by SSL using a gskkyman keyring or a RACF keyring. If the client certificate was issued by a well-known Certificate Authority, it is likely the CA certificate is already primed in the gskkyman keyring. The CA certificate is probably also in RACF. However, all CA certificates in RACF initially have a status of NOTRUST. The CA certificate must be set to TRUST and connected to the appropriate RACF keyring. Refer to "Update CA certificates to TRUST status" on page 735 for detailed information. If the certificate issuer (a CA or self-signed) is not part of the list of well-known CAs, the keyring must be primed with the signer certificate of the CA or the self-signed client certificate.

Once Level 1 authentication is done by SSL using either keyring, the certificate is passed to the server which accesses the RACF database for Level 2 and Level 3 authentication.

Encryption algorithms

Once authentication is done, the client and server must agree on a symmetric encryption method and generate a single encryption key to use for data encryption. The agreed-on key is exchanged using the PKI method of encryption. Once the symmetric encryption algorithm (such as DES) and a single encryption key are chosen, all data exchanges use this algorithm and key instead of the PKI method of encryption.

In an SSL-encrypted session, all data is encrypted using the symmetric encryption algorithm immediately before it is sent to the client. Data from the client is decrypted immediately after it is received. The encryption algorithm that is used for the connection depends on a combination of the encryption algorithm list the SSL subsystem supports, the list the server wants to use, and the encryption algorithms the client requests. During the SSL handshake the client sends a list of encryption algorithms it is willing to use. The server submits its list and the SSL subsystem picks an algorithm all parties support giving preference to the order specified by the server. If the server does not support any of the encryption algorithms requested by the client, the connection is closed. The Telnet, FTP and DCAS servers and the FTP client use the SSL support provided by the System Secure Sockets Layer (System SSL) element of z/OS. The encryption algorithms supported by the servers and client are therefore dependent on the level of System SSL installed. The following encryption algorithms are supported by the base level of System SSL: NULL, RC2 export, RC4 export, DES. The System SSL Level 3 feature is required for Triple DES and RC4 non-export (128 bit) encryption algorithms. The encryption algorithm list can be customized for the servers and client to a subset of the System SSL list. Refer to the security information in the appropriate server or client chapters for specific server and client statements used for encryption list creation.

Encryption is provided either by BSafe software shipped with System SSL or by hardware. There is no TCP profile definition that controls whether the cryptographic hardware will be used for secure connections. When SSL initialization has completed, System SSL checks if ICSF is installed and active and if the hardware is enabled and loaded with the necessary Master Keys. If the hardware is not available at that time, all subsequent encryption is performed using software. If hardware is valid and ICSF is active at that time, the public key functions required during the SSL handshake and requests for encryption using DES and Triple-DES algorithms will be sent to the hardware. Otherwise, all cryptographic functions will be performed by software. Encryption requests using RC2 or RC4 algorithms are always performed by software. Also note that if ICSF subsequently becomes unavailable, System SSL will assume the hardware encryption is still wanted and encryption processing using DES or Triple-DES algorithms will fail until access to the hardware is restored. If subsequent session handshakes are attempted, they will also fail. Completion of SSL initialization is different for each server and client. Refer to the the security information in the appropriate server or client chapters to understand when SSL initialization is complete and how to refresh SSL.

If hardware encryption is to be used, be sure that the RACF user ID associated with the server has read access to the RACF CSFSERV class resources. If ICSF is available but the server has not been given access to these resources, the SSL initialization may fail. The reason code is likely to be 4 (bad password) because System SSL will attempt to use the hardware encryption during processing of the keyring.

Enable CSFSERV resources

If hardware encryption and ICSF are installed, system SSL verifies that the user ID associated with the server is permitted to use CSFSERV resources. The RACF administrator should permit the RACF user ID to use the CSFSERV resources described here.

```
PERMIT service-name CLASS(CSFSERV) ID(serverid) ACCESS(READ)
```

The following CSFSERV resources (service-names) are accessed by System SSL.

1. CSFCKI Clear Key Import
2. CSFCKM Clear Key Import Multiple

3. CSFDEC DES and TripleDES Decipher
4. CSFENC DES and TripleDES Encipher
5. CSFOWH MD5 and SHA1 Hashing
6. CSFRNG Random Number Generate
7. CSFPKB RSA Key Token Build
8. CSFPKX RSA Public Key Extrace
9. CSFPKE RSA Public Key Encipher
10. CSFPKD RSA Private Key Decipher
11. CSFPKI RSA Key Import
12. CSFDSG Digital Signature Generate
13. CSFDSV Digital Signature Verify

The MAXLEN installation option for hardware cryptographic determines the maximum length that can be used to encrypt and decrypt data using ICSF/MVS. Set the MAXLEN ICSF/MVS installation option to 65527 or greater because this is the maximum TCP/IP packet size.

The System SSL GSKSRVR server provides the capability to determine whether cryptographic hardware is being used through its DISPLAY CRYPTO operator command (for example, f gsksrvr,d crypto). The System SSL GSKSRVR server is not automatically started. For additional information on setting up and using the GSKSRVR server, refer to *z/OS System Secure Sockets Layer Programming*.

Refer to *z/OS ICSF Administrator's Guide* for additional information on controlling who can use cryptographic keys and services.

Creating and managing keys and certificates at the server

Overview

The gskkyman utility or RACF Common Keyring support can be used to manage the keys and certificates needed for SSL support.

The following table describes the steps necessary to implement the different levels of SSL security for each keyring management product.

		Keyring management product	
SSL function	**Steps**	**gskkyman**	**RACF**
Server Auth	1. Create a keyring file 2. Create a server certificate **If server certificate is self-signed** 3. Extract server certificate from server keyring 4. Add server certificate to client keyring	1. Page 729 2. Page 730 3. Page 733 4. Page 744	1. Page 735 2. Page 736 3. Page 736 4. Page 744
Client Auth Level 1	1. Set up server authentication **If client certificate is self-signed** 2. Extract client certificate from client keyring 3. Add client certificate to server keyring	1. See above 2. Page 741 3. Page 733	1. See above 2. Page 741 3. Page 733

		Keyring management product	
SSL function	**Steps**	**gskkyman**	**RACF**
Client Auth Level 2	1. Set up server authentication 2. Set up level 1 authentication 3. Associate certificate to RACF User ID	1. See above 2. See above 3. Page 737	1. See above 2. See above 3. Page 737
Client Auth Level 3	1. Set up server authentication 2. Set up Level 1 authentication 3. Set up Level 2 authentication 4. Add User IDs to the server's SERVAUTH profile access list	1. See above 2. See above 3. See above 4. Page 737	1. See above 2. See above 3. See above 4. Page 737
Express Logon Feature	1. Set up server authentication 2. Set up level 1 authentication (optional for DCAS) 3. Set up Level 2 authentication (optional for DCAS) 4. Define passticket profiles	1. See above 2. See above 3. See above 4. Page 738	1. See above 2. See above 3. See above 4. Page 738

Certificate file types

The following sections mention several certificate formats. Below is a high level summary of the differences.

- PKCS12 files are used to move the server certificate to another server keyring. Because this format contains the private key, the file is usually password protected. IBM recommends only using this format when required.
  - Commonly used file extension is .p12
  - Contains private key, public key and certificate
  - Created by
    - HOD export function

      When the HOD client specifies a client certificate to send to the server during SSL processing, it must be in this format. The private key portion is not sent to the server.
    - Netscape export function
    - gskkyman ″Export keys to a PKCS12 file″ function
- Certificate files are normally needed when a self-signed certificate is used. In this case, each self-signed certificate appears to be signed by a unique CA. Therefore, the client's keyring (if this is a self-signed server certificate) or server's keyring (if this is a self-signed client certificate) must be primed to recognize the issuer of the self-signed certificate. This format can be used to prime a keyring with the issuer's CA certificate. This format can also be used when registering a client certificate with RACF.
  - Commonly used file extensions are .crt and .der
  - Contains public key and certificate
  - Created by
    - HOD extract function
    - gskkyman's ″Create a self-signed certificate″ function

Common terminology

The following variable names are used throughout the appendix:

tnserverid
The user ID defined to RACF given superuser status that represents the TCPIP stack and the TN3270 Server.

dcasserverid
The user ID defined to RACF given superuser status that represents the Digital Certificate Access Server.

ftpserverid
The user ID defined to RACF given superuser status that represents the FTP daemon and all spawned FTP Servers.

serverid
The user ID defined to RACF given superuser status that represents any of the servers above.

userid The user ID that is associated with a client certificate in the RACF keyring database. Or the TSO user ID that requires authority to issue certain RACF commands.

Copying HFS files to MVS data sets

Certificate and database files are often stored in HFS file formats and sometimes need to be copied into MVS file formats. MVS files can be created from HFS files by using the TSO OGET command with the BINARY option and can be protected using RACF. For example:

```
OGET '/tmp/telnet/mvs180.kdb' 'TCPCS6.MVS180.KDB' BINARY
OGET '/tmp/telnet/mvs180.sth' 'TCPCS6.MVS180.STH' BINARY
```

It is recommended that the MVS dsnames be the HFS filenames prefixed by one or more high-level qualifiers. The same high-level qualifier(s) must be used for both the keyring and the stash file. This ensures that the name relation used to generate the stash file from the keyring file is unchanged. Refer to *z/OS UNIX System Services Command Reference* for more information on the use of the OGET command.

Using the gskkyman utility

This section gives examples of how to use the gskkyman utility to:

- Create keyrings
- Create a server self-signed certificate
- Extract a server certificate
- Add client certificates into a keyring

The gskkyman utility is a command-line utility. It prompts you for the information you need to perform a task. If you make an error, it issues a message and prompts you again for the information.

The gskkyman utility is documented in *z/OS System Secure Sockets Layer Programming*. It is recommended that you read the gskkyman topics in this document before starting to use the gskkyman.

Additional information and examples can also be found in the following Redbooks:

- *SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements*
- *IBM SecureWay Host On-Demand: Enterprise Communications in the Era of Network Computing*

To run gskkyman, you must have access to the z/OS Cryptographic Services message catalogs and DLLs. The C DLL Library (that is, SYS1.SCLBDLL) must be available and APF authorized to avoid possible abends caused by trying to access a nonexisting or non-APF authorized system. For example, if the z/OS Cryptographic Service DLL library is not part of the linklist concatenation, an ″export STEPLIB=hlq.SGSKLOAD″ command might be needed. For additional information, refer to *z/OS System Secure Sockets Layer Programming*.

The gskkyman utility is shipped with z/OS in System SSL as a part of the Cryptographic Services Base element of z/OS. It supports the generation of key sizes of 1024 or 2048 bits. Note that if you have existing keys with a size of 512, these keys are still usable. The gskkyman utility runs under the z/OS shell and can create several types of HFS files. System SSL requires the following files:

- A keyring file (also known as a key database).
- A password file (also known as a stash file) which contains the password associated with the keyring file.

The keyring file and the stash file are used by the server to obtain the server's certificate and the public/private key pair used during SSL handshake processing. The server uses the stash file as the mechanism to obtain the keyring password rather than using a configuration parameter which might be accessible to a larger number of people. The stash file is created by using gskkyman's 'Store encrypted database password' function on the main menu.

Security of these files is an installation responsibility. It is recommended to restrict the file access to users with superuser authority.

The server must have read and write access to the key database and read access to the password file.

Note: The gskkyman utility accepts only the HFS files.

The gskkyman utility allows you to enter the fully qualified path and file name when it prompts you for a keyring, certificate request, or certificate file name. However, you should change to the path where the file will be stored before you start gskkyman.

Create a keyring file

Before starting gskkyman, we suggest that you start in the directory where the keyring is to be created. Otherwise, be sure to include the full file name when specifying the keyring name.

1. Start gskkyman. This will display the *Database Menu*. Select *Create new database* (option 1).
2. Specify the key database name, password, and expiration information as requested.
3. Create a password file (also known as a stash file) by selecting *Store database password* (option 10) from the *Key Management Menu*.

Following is a sample of the gskkyman output for creating a key database. Sample user replies are shown in **bold** characters.

1. The *Database Menu* that starts the process for creating a key database is shown below. Select the *Create new database* option.

```
      Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database

   0 - Exit program

Enter option number: 1
```

2. Specify the key database name, password, and expiration information as requested.

```
Enter key database name (press ENTER to return to menu): server.kdb
Enter database password (press ENTER to return to menu):
Re-enter database password:
Enter password expiration in days (press ENTER for no expiration): 365
Enter database record length (press ENTER to use 2500):

Key database /SYSTEM/usr/keyring/server.kdb created.

Press ENTER to continue.
```

3. Pressing enter brings up the *Key Management Menu* as follows. Create the password file by selecting the *Store database password* option.

```
     Key Management Menu

   Database: /SYSTEM/usr/keyring/server.kdb

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive certificate issued for your request
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 10
Database password stored in /SYSTEM/usr/keyring/server.sth.

Press ENTER to continue.
```

Pressing enter returns you to the *Key Management Menu*. You can proceed to create your server certificate or exit.

Create a server self-signed certificate

Refer to gskkyman documentation for the steps necessary to create a CA-signed server certificate. With gskkyman, you can create your own self-signed certificate for testing:

1. From the *Database Menu*, open your keyring file (Option 2–*Open database*). From the *Key Management Menu*, select *Create a self-signed certificate* (option 6).

2. Specify the certificate type to be created from one of the end user certificate options. The type of end user certificate created depends on the security requirements of your installation.
3. Specify the label, subject name, and length of time the certificate is valid as requested.
4. After the certificate is created, set the certificate as the default by selecting *Manage keys and certificates* (option 1), selecting the certificate you created, followed by selecting *Set key as default* (option 3).

The following is a sample of the gskkyman output for creating a self-signed certificate. Sample user replies are shown in **bold** characters.

1. The *Key Management Menu* that starts the process for creating a self-signed certificate is shown below. Select *Create a self-signed certificate* (option 6).

```
     Key Management Menu

     Database: /SYSTEM/usr/keyring/server.kdb

 1 - Manage keys and certificates
 2 - Manage certificates
 3 - Manage certificate requests
 4 - Create new certificate request
 5 - Receive certificate issued for your request
 6 - Create a self-signed certificate
 7 - Import a certificate
 8 - Import a certificate and a private key
 9 - Show the default key
10 - Store database password
11 - Show database record length

 0 - Exit program


Enter option number (press ENTER to return to previous menu): 6
```

2. The *Certificate Type* menu will be displayed. Select one of the end user certificate types (options 4, 5, or 6).

```
       Certificate Type

   1 - CA certificate with 1024-bit RSA key
   2 - CA certificate with 2048-bit RSA key
   3 - CA certificate with 1024-bit DSA key
   4 - End user certificate with 1024-bit RSA key
   5 - End user certificate with 2048-bit RSA key
   6 - End user certificate with 1024-bit DSA key

Select certificate type (press ENTER to return to menu): 4
```

3. You will then be asked for the certificate data and the certificate will be created.

```
Enter label (press ENTER to return to menu): selfsignedcert
Enter subject name for certificate
  Common name (required): test server certificate
  Organizational unit (optional): dev
  Organization (required): dev
  City/Locality (optional):
  State/Province (optional):
  Country/Region (2 characters - required): US
Enter number of days certificate will be valid (default 365): 5000

Please wait .....

Certificate created.

Press ENTER to continue.
```

4. Make the self-signed certificate the default server certificate. Pressing enter returns you to the *Key Management Menu* shown in step 1 above. Select *Manage keys and certificates* (option 1). This will bring up the *Key and Certificate List* menu shown below. Select the number that corresponds to the self-signed certificate just created.

```
       Key and Certificate List

       Database: /SYSTEM/usr/keyring/server.kdb

   1 - selfsignedcert

   0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list): 1
```

   This will bring up the *Key and Certificate Menu*. Select *Set key as default* (option 3).

```
       Key and Certificate Menu

       Label: selfsignedcert

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 3

Default key set.
```

To use the new default server certificate, the server must reinitialize its SSL environment. Refer to the security information in the appropriate server or client chapters for SSL initialization details. The self-signed server certificate will need to be added to the client's key database as a CA. For additional information, see "Extract the server certificate from the keyring" on page 733.

Extract the server certificate from the keyring

If using a self-signed server certificate, the server certificate must be added to the client's key database as a CA certificate. Some clients provide the capability to extract the server's certificate when the client connects to the server. For some clients, however, this process must be done manually by exporting the server's certificate to a file, sending the server's certificate file to the client, and adding the server's certificate to the client's key database.

The server certificate can be exported to a file from the *Key and Certificate Menu*. If you have just opened your keyring file and are at the *Key Management Menu*, select *Manage keys and certificates* (option 1) and select the certificate you want to export from the list. You will then see the *Key and Certificate Menu*.

1. Select *Export certificate to a file* (option 6).
2. Specify the desired format and file name.

The following is a sample of the gskkyman output for exporting a certificate. Sample user replies are shown in **bold** characters.

1. The *Key and Certificate Menu* that starts the export process follows. Select *Export certificate to a file* (option 6).

```
       Key and Certificate Menu

       Label: selfsignedcert

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 6
```

2. The *Export File Format* menu is displayed, and you will be asked to select the export format and to enter a file name.

```
       Export File Format

   1 - Binary ASN.1 DER
   2 - Base64 ASN.1 DER
   3 - Binary PKCS #7
   4 - Base64 PKCS #7

Select export format (press ENTER to return to menu): 1
Enter export file name (press ENTER to return to menu): binservercert.der

Certificate exported.
```

   If using ftp to send the export file to the client, remember to send in binary format if the file format chosen above was binary.

Add client certificates to the server keyring

If using a client self-signed certificate, the certificate must be added to the server keyring as a CA certificate. Send the client certificate to the MVS host using FTP

(with the BINARY send option if the certificate was extracted in binary format). The client may be the DCAR, FTP client, or TN3270 client. Use gskkyman's 'Store a CA certificate' option to obtain the client CA from the binary DER file. If the client certificate was issued by a well-know CA, only the signer's certificate needs to be in the keyring. The gskkyman utility primes its keyrings with several well-known CAs.

Using RACF's common keyring support

This section gives examples of how to use RACF common keyring support to:

- Create keyrings
- Create a server self-signed certificate
- Extract a server certificate
- Add client certificates into a keyring

RACF can be used to manage the keys and certificates normally stored in the key database. All the functions that the gskkyman utility provides, are also available in the RACF support. However, because RACF can manage multiple keyrings, certificates and keyrings are added independently. A certificate is then connected to one or more keyrings.

All the server keyrings and certificates are stored in the RACF database. There are no separate key database or stash files.

Refer to *z/OS Security Server RACF Security Administrator's Guide* for information about how to use RACF to manage your key database information.

For detailed information on the RACDCERT command and other commands that might be useful in managing your RACF keyring data, refer to *z/OS Security Server RACF Command Language Reference* for the full syntax and description of these commands.

RACF keyring names, labels, and so on are case-sensitive. When adding the keyring name to the server profile, be sure that the correct case is used.

Before using RACF to store your key database information:

1. Ensure that RACDCERT is defined as an authorized TSO command in the IKJTSOxx member.
2. The well-known CA certificates are initially marked as NOTRUST in the RACF database and you will have to update the CA certificates that you plan to support to TRUST status.
3. Refresh the applicable RACF class after making changes.
4. There are various ways to register the client certificate with RACF or set up a RACF Certificate Name Filter. For a complete description of RACF management of digital certificates and options available, see the *z/OS Security Server RACF Security Administrator's Guide*. If using RACF's Certificate Name Filtering with MultiID filters, TN3270 client authentication processing only matches filters that specify generic (*) criteria.
5. Most tasks related to certificates are managed using the RACDCERT command. The issuer of these commands must have appropriate RACF authority to the IRR.DIGTCERT.function resource in the FACILITY class with UPDATE, CONTROL, or READ ACCESS. Refer to the *z/OS Security Server RACF Security Administrator's Guide* for more information on controlling the use of the RACDCERT command and for a complete description of functions needed for keyring and certificate use.

6. Consider RACLISTing the DIGTCERT class for best performance.

RACF panels also support most of the certificate and keyring functions and can be used to perform these actions, if desired.

Configuring RACF services for the servers

This section describes some commands needed for configuring RACF for the servers. These commands are in EZARACF in the SEZAINST data set.

Update CA certificates to TRUST status: Several well-known CA certificates are primed in the RACF database but are initially marked NOTRUST. To use the certificate, it must be changed to TRUST status. As an example, the commands below show how to change the Verisign Class 3 CA to trusted status and then connect it to a keyring.

```
RACDCERT CERTAUTH ALTER( LABEL('Verisign Class 3 Primary CA')) TRUST

RACDCERT ID(TCPID) CONNECT (CERTAUTH    RING(SERVERKeyring)
      LABEL('Verisign Class 3 Primary CA') USAGE(CERTAUTH)  )
```

Activate the DIGTCERT, DIGTRING, and optional DIGTNMAP classes: The DIGTCERT and DIGTRING classes must be active before defining certificates or keyrings to RACF by using the SETROPTS commands. For example:

```
SETROPTS CLASSACT(DIGTCERT)
SETROPTS CLASSACT(DIGTRING)
```

If using Certificate Name Filtering, ensure that the DIGTNMAP class is active. For example:

```
SETOPTS CLASSACT(DIGTNMAP)
```

Also be sure to do a refresh after any changes. For example :

```
SETROPTS RACLIST(DIGTCERT) REFRESH
SETROPTS RACLIST(DIGTRING) REFRESH
SETROPTS RACLIST(DIGTNMAP) REFRESH
```

Allow SSL keyring confirmation: System SSL verifies the server RACF user ID does have access to the keyring. Therefore, if the server is started as an MVS started procedure, you must permit the RACF user ID associated with the server to have control access to the IRR.DIGTCERT.LISTRING. For example:

```
RDEFINE FACILITY  (IRR.DIGTCERT.function )  UACC(NONE)
PERMIT  IRR.DIGTCERT.LISTRING  CLASS(FACILITY)  ID(serverid) ACCESS(CONTROL)
```

If the DCAS is started from a TSO user ID under the z/OS UNIX shell, you must also permit that ID. For example:

```
RDEFINE FACILITY  (IRR.DIGTCERT.function )  UACC(NONE)
PERMIT  IRR.DIGTCERT.LISTRING  CLASS(FACILITY)  ID(userid) ACCESS(CONTROL)
```

Create a keyring file

Use the following RACF command to add (create) a server keyring and associate it with the server RACF user ID:

```
RACDCERT ID(serverid) ADDRING(SERVERRING)
```

To delete or list a keyring:

```
racdcert ID(serverid) delring(SERVERRING)
racdcert ID(serverid) listring(SERVERRING)
```

Create a server self-signed certificate

Refer to the RACF documentation for the steps necessary to create a CA-signed server certificate. To create a self-signed server certificate called XXXDN, for user ID serverid, use the following command, where CN is the common name and OU is the organization unit name. Additional options are available within SUBJECTSDSN.

```
RACDCERT ID(serverid)  GENCERT  SUBJECTSDSN(CN('UNIT1') OU('TESTING') C('US') ) TRUST
  WITHLABEL('XXXDN') SIZE(1024)
```

To connect the certificate to a keyring and make it the default certificate, use the following command. This example assumes a keyring called serverRing has already been created.

```
RACDCERT ID(serverid) CONNECT(ID(serverid)  LABEL('XXXDN') RING(SERVERRING) DEFAULT)
```

Extract a server certificate from a server keyring

If using FTP to send the server self-signed certificate to the client, use RACDCERT Export to export the server certificate to an MVS file in DER format. The server self-signed certificate must be added to the client keyring to prime it for decrypting the server certificate. EXPORT generally implies exporting both a certificate and private key. However, the CERTDER format instructs the command to export only the certificate in DER format, which is generally considered an EXTRACT. Use the following RACF command:

```
RACDCERT ID(serverid) EXPORT(LABEL('XXXDN')) DSN('dataset name') FORMAT(CERTDER)
```

Add client certificates to the server keyring

If using a client self-signed certificate, the certificate must be added to the server keyring as a CA certificate. Send the client certificate to the MVS host using FTP (with the BINARY send option). The client may be the DCAR, FTP client, or TN3270 client. If the client certificate is issued by a well-known CA, only the signer's certificate needs to be in the keyring. The well-known CA certificates are initially marked as NOTRUST and must be updated to TRUST status.

The client self-signed certificate must be registered into the RACF database before the certificate can be associated with a keyring. Associate the certificate to a RACF user ID to register the certificate into RACF. For example:

```
RACDCERT ID(USER2) ADD('SSCLNTCERT.USER2.DER') WITHLABEL('CLNTCERT_USER2') TRUST
```

This command requires that the certificate be defined in an MVS data set. If the certificate is defined in the HFS, you can use the TSO OGET command (with the BINARY send option) to move the certificate to an MVS data set.

Use the RACDCERT CONNECT command to connect the client certificate to the RACF keyring as a CA certificate. In this example, the RACF user ID associated with the server is serverid and the keyring name used by the server is serverRing:

```
RACDCERT ID(serverid) CONNECT (ID(USER2) RING(serverRing)
      LABEL('CLNTCERT_USER2') USAGE(CERTAUTH) )
```

Associate certificate with user ID

Use the RACDCERT ADD command to register the client's certificate and associate it with a user ID. In this example, the binary DER client certificate has been stored in an MVS file named 'SSCLNTCERT.USER2.DER', and is to be associated with the RACF user ID USER2, and given a label 'CLNTCERT_USER2':

```
RACDCERT ID(USER2) ADD('SSCLNTCERT.USER2.DER') WITHLABEL('CLNTCERT_USER2') TRUST
```

Be sure to refresh the DIGTCERT and DIGTRING class. For example:

```
SETROPTS RACLIST (DIGTRING) REFRESH
SETROPTS RACLIST (DIGTCERT) REFRESH
```

Reinitialize SSL: Reinitialize the server SSL to pick up the certificates that have been added to the key database. Refer to the security information in the appropriate server and client chapters to understand when SSL initialization is complete and how to refresh SSL.

Add user IDs to the SERVAUTH profile access list

With level 3 authentication, you specify the user IDs that are allowed to connect into a specific Telnet port, DCAS server, or FTP port by associating the user IDs to each server's RACF SERVAUTH profile. Refer to the security information in the appropriate server and client chapters for level 3 setup. The user ID associated with the client certificate can then be checked against the SERVAUTH class profile entry. The use of this RACF class is optional. If the SERVAUTH RACF class is active and a RACF profile for the port is defined, this level of RACF authorization will be verified prior to connection negotiation. If the SERVAUTH class is not active or there is no RACF profile, this indicates that this level of check is not required and the client is allowed to connect to the server as long as the client certificate was validated.

TN3270 server: The RACF profile name is:

```
EZB.TN3270.sysname.tcpname.PORTnnnnn
```

where nnnnn is the port number with leading zeros. The profile name can contain wildcards to the extent that the security product allows. All security product rules (for example wildcards, PROTECTALL, and so on) apply. For example, the profile name for TCP stack TCPCS running on system MVSA for port 992 would be:

```
EZB.TN3270.MVSA.TCPCS.PORT00992
```

If all systems will use the same access list, and RACF generic profile checking is active for the SERVAUTH class, the following profile name could be used:

```
EZB.TN3270.*.TCPCS.PORT00992
```

To protect all ports with a single profile, the following security product profile name could be used:

```
EZB.TN3270.MVS.TCPCS.PORT*
```

To restrict access on a port basis, the following RACF setup is needed and must be done by a user ID that has authority to issue the specified RACF commands:

- Activate the RACF SERVAUTH class, if not active:

  ```
  SETROPTS CLASSACT(SERVAUTH)
  ```

- Define the profile for the Telnet port:

  ```
  RDEFINE  SERVAUTH   EZB.TN3270.sysname.tcpname.PORTnnnnn  UACC(NONE)
  ```

- Permit the user ID associated with TCP to the port profile:

  ```
  PERMIT EZB.TN3270.sysname.tcpname.PORTnnnnn CL(SERVAUTH) ID(tcpuserid) ACCESS(READ)
  ```

- Ensure the SERVAUTH class is RACLISTed. If it is not, RACLIST it:

  ```
  SETROPTS RACLIST(SERVAUTH)
  ```

- Refresh the SERVAUTH class before using:

  ```
  SETROPTS RACLIST(SERVAUTH) REFRESH
  ```

DCAS: The RACF profile name is:

```
EZA.DCAS.cvtsysname
```

- Activate the RACF SERVAUTH class, if not active:

  ```
  SETROPTS CLASSACT(SERVAUTH)
  ```

- Define the profile:

```
RDEFINE SERVAUTH EZA.DCAS.cvtsysname UACC(NONE)
```

- Permit the user ID associated with TCP to the port profile:

```
PERMIT EZA.DCAS.cvtsysname CLASS(SERVAUTH) ACCESS(CONTROL) ID(dcasid)
```

- Ensure the SERVAUTH class is RACLISTed. If it is not, RACLIST it:

```
SETROPTS RACLIST(SERVAUTH)
```

- Refresh the SERVAUTH class before using:

```
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Note: The RACF user ID associated with the certificate and the EZA.DCAS.cvtsysname can be any valid user ID.

FTP server: The RACF profile name is:

```
EZB.FTP.sysname.ftpdaemonname.PORTnnnn
```

where nnnnn is the port number with leading zeros. The profile name can contain wildcards to the extent that the security product allows. All security product rules (for example wildcards, PROTECTALL, and so on) apply. For example, the profile name for FTP daemon FTPD running on system MVSA for port 992 would be:

```
EZB.FTP.MVSA.FTPD.PORT00992
```

If all systems will use the same access list and RACF generic profile checking is active for the SERVAUTH class, the following profile name could be used:

```
EZB.FTP.* .FTPD.PORT00992
```

To protect all ports with a single profile, the following security product profile name could be used:

```
EZB.FTP.MVS.FTPD.PORT*
```

To restrict access on a port basis, the following RACF setup is needed and must be done by a user ID that has authority to issue the specified RACF commands:

- Activate the RACF SERVAUTH class, if not active:

```
SETROPTS CLASSACT(SERVAUTH)
```

- Define the profile for the FTP port:

```
RDEFINE  SERVAUTH   EZB.FTP.sysname.ftpdaemonname.PORTxxxxx  UACC(NONE)
```

- Permit the user ID associated with the FTP daemon to the port profile:

```
PERMIT EZB.FTP.sysname.ftpdaemonname.PORTnnnnn CL(SERVAUTH)   ID(tcpuserid)  ACCESS(READ)
```

- Ensure the SERVAUTH class is RACLISTed. If it is not, RACLIST it:

```
SETROPTS RACLIST(SERVAUTH)
```

- Refresh the SERVAUTH class before using:

```
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Define PassTicket profiles to RACF

The following recommendations apply when defining PassTicket profiles to RACF:

- Use the SETROPTS CLASSACT(PTKTDATA) command to activate the PTKTDATA class.
- Use the SETROPTS RACLIST (PTKTDATA) command to RACLIST the PTKTDATA class.
- For each application to which users want to gain access with a PassTicket, you must define a PTKTDATA class profile. For example, to give users access to TSO, define a profile for TSO using the following command:

```
RDEFINE PTKTDATA  TSO3050
                    SSIGNON(KEYMASKED())
                    UACC(NONE)
```

The DIGTNMAP and DIGTCRIT classes support profiles. The application ID used for DIGTCRIT profiles must be the same as that used on the HOD V5 Application ID popup window.

Defining profiles for applications such as TSO can be tricky because RACF has special methods for naming profiles. For more information, refer to the *z/OS Security Server RACF Security Administrator's Guide*.

Add the client certificate to the RACF server keyring and associate the client certificate with a user ID.

Define a PassTicket profile for each application accessed by Express Logon. The application ID portion of the profile must match that configured on the HOD V5 workstation Application ID popup window.

```
SETR  CLASSACT(PTKTDATA)        //* Activate the PassTicket data class
SETROPTS  RACLIST(PTKTDATA)     //* Raclist the class
SETR RACLIST(PTKTDATA) REFRESH
RDEFINE   PTKTDATA  TSO3390  SSIGNON()  UACC(NONE)
```

Migrating an existing gskkyman key database to RACF

To migrate from an existing key database (kdb) created by gskkyman, each certificate that the customer has added must be individually exported and then added to the RACF database. The RACF database is already primed with some well-known Certificate Authorities (CA), so it is not necessary to migrate these CA certificates to RACF. Note however, that the well-known CAs are initially marked as NOTRUST in the RACF database and you will have to update the CA certificates that you plan to support to TRUST status.

To migrate a server certificate from your kdb created by gskkyman to RACF:

1. Use gskkyman to export the certificate and key to a PKCS12 format file:
   a. Open the key database file that you want to migrate.
   b. Select 'List/Manage keys and certificates'.
   c. Select the certificate to be exported.
   d. Select 'Export the certificate and key to a file', supply the name of the HFS file where the certificate and key will be stored (in PKCS12 format) and enter the password to protect the file when prompted.
2. Use the OGET command described in "Copying HFS files to MVS data sets" on page 728 to create an MVS file.
3. Add the certificate and key to the RACF database and assign it to a user, if applicable. If this is the server certificate, assign it to the user ID associated with the server. For example:

   ```
   RACDCERT ID(serverid) ADD('Serverid.mycert.p12') WITHLABEL('ServerCert')
   PASSWORD('mypw') TRUST
   ```

You will also need to create a keyring for your server. For example:

```
RACDCERT ID(serverid) ADDRING(serverring)
```

Then connect the appropriate certificates to the keyring. For example, to connect the default server certificate that was migrated above ('ServerCertificate') to the 'serverring' that we associated with serverid:

```
RACDCERT ID(serverid) CONNECT( ID(serverid) LABEL('ServerCert')
RING(ServerRing) DEFAULT )
```

Creating and managing keys and certificates at the client

Normally, a client certificate should be obtained from a well-known Certificate Authority (CA). The Certificate Authority's root certificate needs to be included in the server's key data base as a trusted authority in order for the client's certificate to pass the SSL protocol's client authentication process. If the client certificate has been issued by a well-known CA, the client certificate need not reside in the server's key database. If an installation uses self-signed client certificates for testing purposes, each certificate appears to be issued by a unique CA. Therefore, each self-signed client certificate must be added to the server's key database as a CA.

If you also want verification that the client certificate is registered with your security product, the client certificate must reside in the server's security product database (using the RACDCERT command with the ADD option is one way to add the client certificate to RACF). Refer to "Using RACF's common keyring support" on page 734 for more information on using RACF to store certificates. If the installation is using self-signed client certificates and requests verification that the client certificate is registered with the security product, the client certificate must reside in both the server's key database (as a CA certificate) and in RACF.

Steps to create a self-signed client certificate vary depending on the source of the client certificate. Refer to "Create a self-signed client certificate" for details on creating a client certificate using HOD's Certificate Management utility.

After the client certificate has been created and extracted as a DER data file at the client, FTP the binary DER data file to the z/OS host using FTP's binary option. If using RACF for the keyring, level 2 client authentication, or level 3 client authentication, an MVS file will need to be created. If FTP created an HFS file, use the OGET command described in "Copying HFS files to MVS data sets" on page 728 to create an MVS file.

Create a self-signed client certificate

See HOD's online documentation for additional details. This sample uses a locally installed HOD V4 client on an NT system using HODs Key Management Utility.

1. On the HOD client, go to HOD's Certificate Management panels (go to Start, Programs, Host On Demand, Administration, Certificate Management.) and open up the key database by selecting the open icon. Usually a key database will exist. If you have never used the key database, it might have a default password (usually ncod - the help menu should contain help information that specifies the default password for your system) or you can select the new option. If new is selected, the correct path and file name will normally be filled in by HOD. Do not change this file name. The HOD key database is normally in HOD's bin subdirectory and named HODClientKeyDb.kdb.

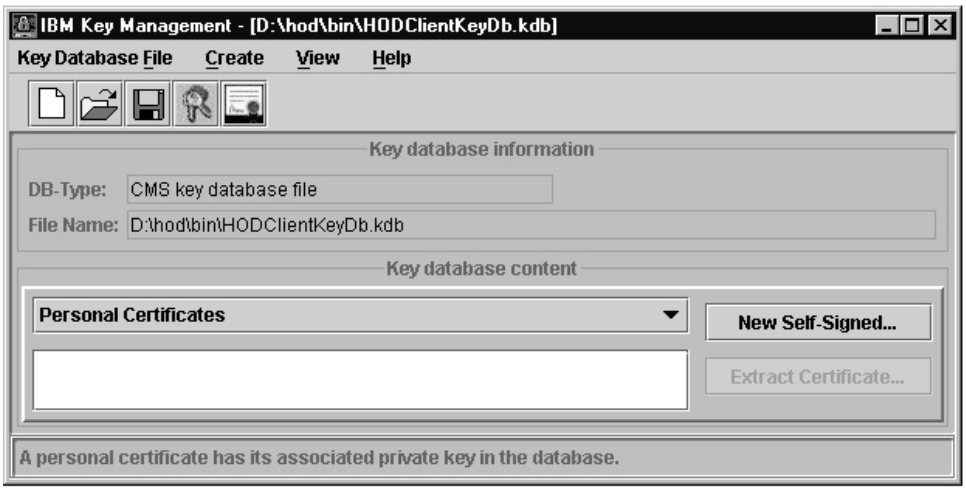


Figure 81. IBM Keys Management

If *Personal Certificates* is not displayed, click the drop-down list arrow and select *Personal Certificates* from the pull-down list.

2. Create a self-signed personal certificate by selecting the *New Self-Signed* button. The *Create New Self-Signed Certificate* screen is displayed.

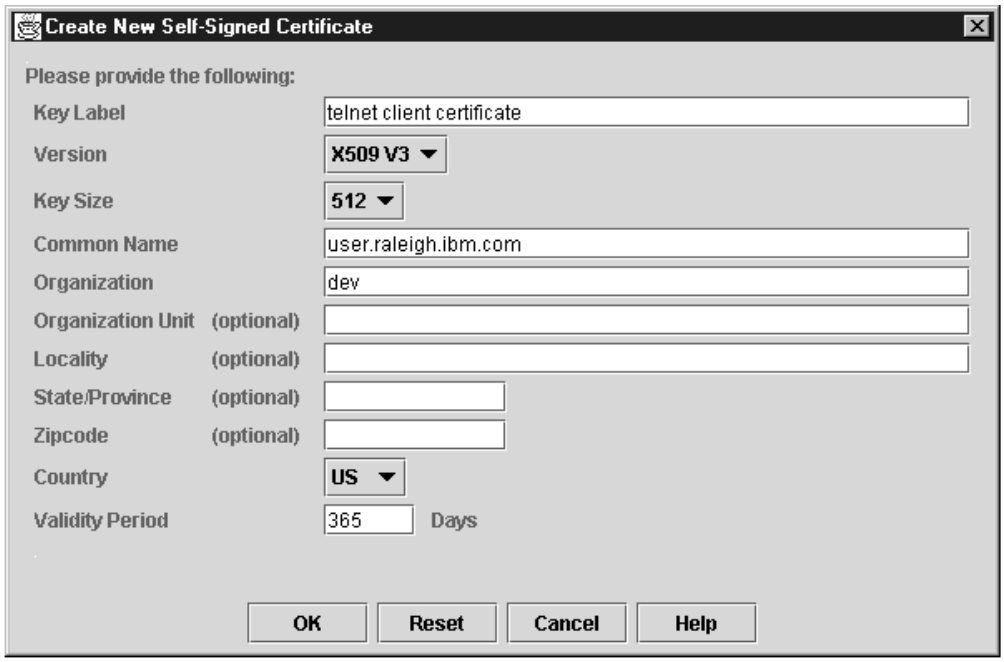


Figure 82. Create New Self-Signed Certificate

Fill in the requested information and then click OK. The new certificates will now be in the personal certificates list.

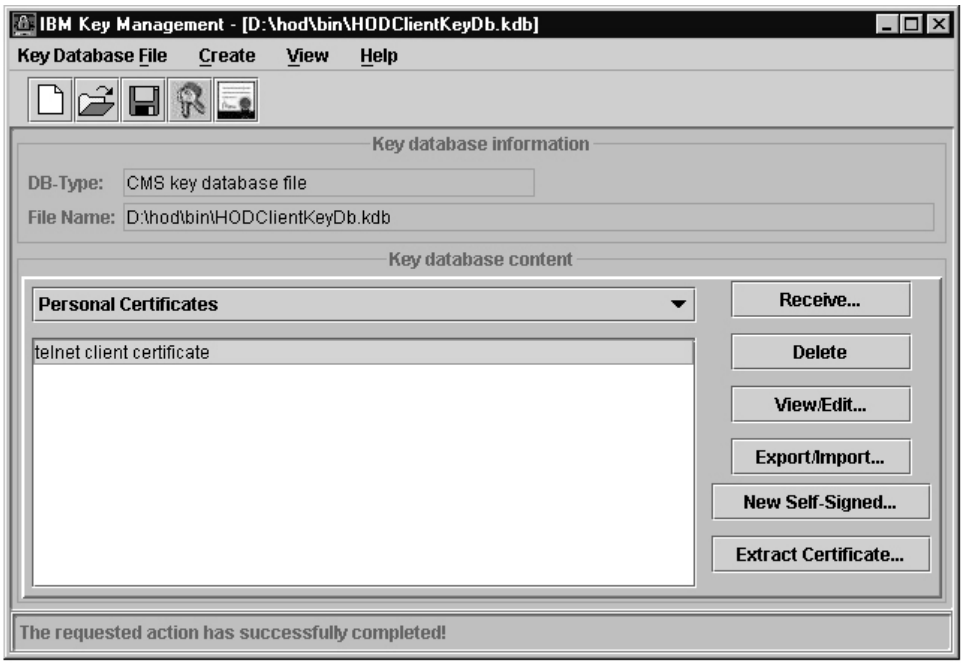


Figure 83. IBM Key Management

3. Use the export function by selecting the Export/Import button to create a PKCS12 file. This is the file that the HOD client will use.

   Specify the path and file where the exported PKCS12 file will be stored and click OK. Enter a password to protect the file when prompted.

Figure 84. Export/Import Key

4. Create the certificate file that will be used to prime the server's keyring with the CA for the self-signed client certificate.

   Use the Extract Certificate function from the panel shown in Figure 83 to create a binary DER data file. This file will have the format filename.der.

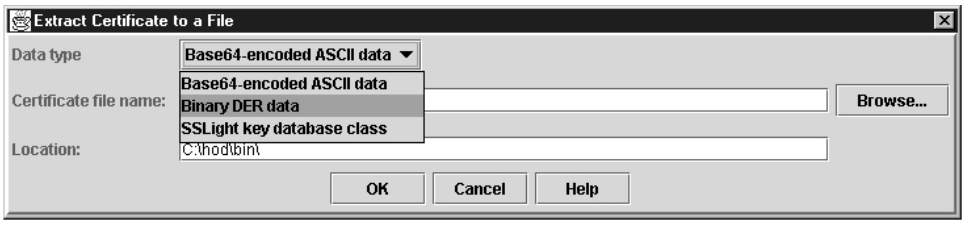


Figure 85. Extract Certificate to a File

Specify the path and file where the exported binary DER file will be stored and click on OK. This is the file you will FTP to the server host.

5. When you start a session from HOD to a port that requires a client certificate, HOD will display a panel that requests the client certificate file and password. The pkcs12 file created in step 3, should be specified.

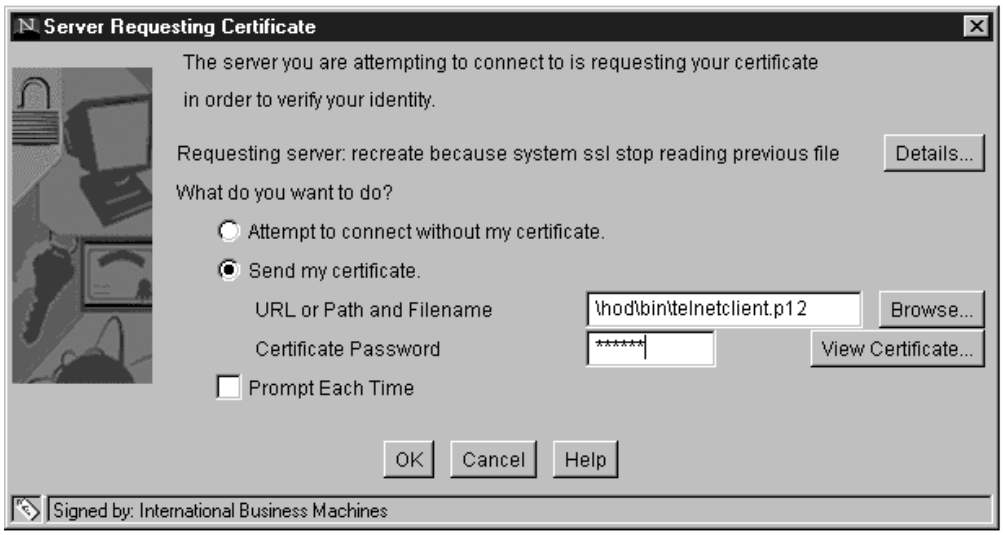


Figure 86. HOD connection using a client certificate

Note: If using HOD and you are connecting to a port that requires a client certificate, the security properties for the connection must indicate that a certificate should be sent. The following example shows the HOD Security properties screen.

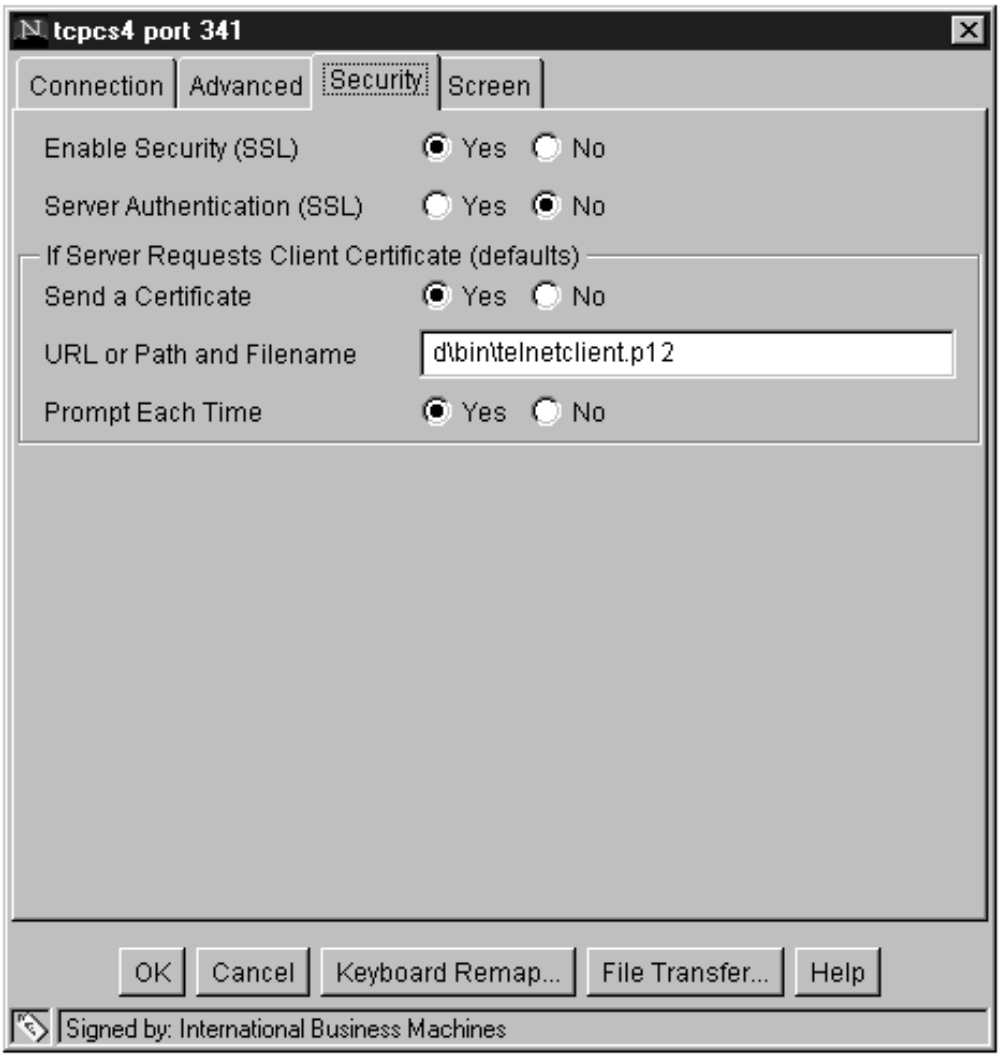


Figure 87. HOD security properties

Add server certificates to the client keyring

1. Get the server certificate information to the client machine.
   - Extract the server certificate from the server keyring. FTP can be used to ship the server certificate file to the client.
   - Newer versions of HOD allow you to extract the server information directly from the HOD client window during connection setup and eliminate the need to FTP the server certificate to the clients. The following is an example of using this method. Once the server side has been configured for the secure port and the port is active:
     – Setup your hod client to connect into the secure port and try the connection.
     – If the connection fails with a 662 (indicating the 'server presented a certificate that was not trusted'), you do not have the certificate of the CA that issued the server certificate in you client's keyring.

- From the client window, select communication from the action bar, then select security. Information for the server certificate should be displayed:

Figure 88. Security Information

- Select extract and indicate binary format and where to store the certificate, then click OK.

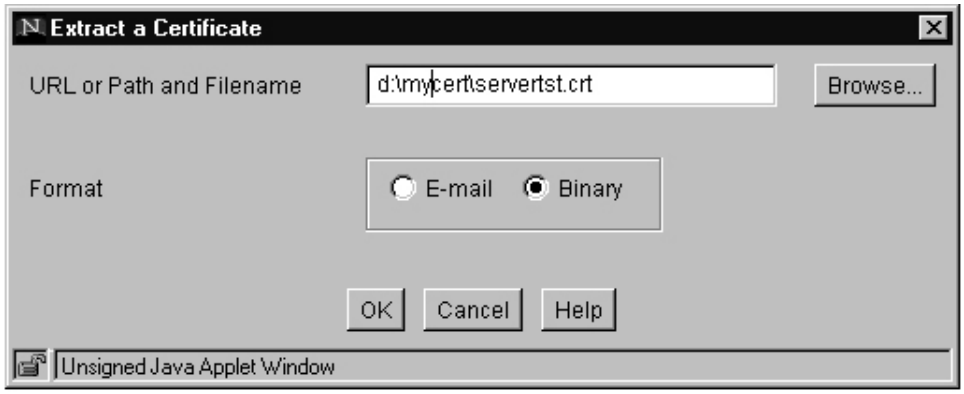


Figure 89. Extract a Certificate

If the action was successful, the following is displayed, which indicates that the server certificate information is on your client system.

Figure 90. Certificate was extracted

2. Add the self-signed server certificate to HOD's CustomizedCAs:
   - On the HOD client, go to HOD's Certificate Management panels. To do this, select Start, Programs, IBM Host On-Demand, Administration, Certificate Management.
   - Open the CustomizedCAs.class file. If customized CA certificates have previously been added to HOD, or if this is the first customized CA, create a new class file by:
     – Selecting File, then New.
     – Click on the Key Database Type arrow and select SSLight key database class. This automatically fills in the required filename and path

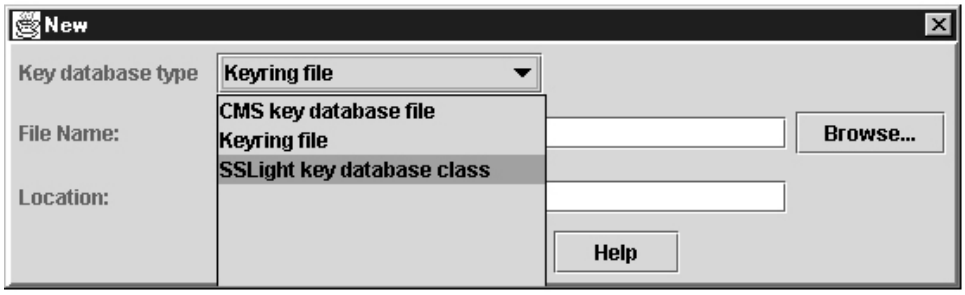


Figure 91. Creating a new CustomizedCAs.class

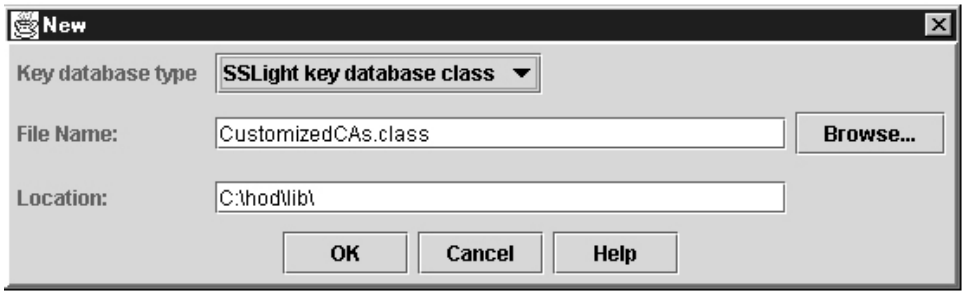


Figure 92. Default location displayed

     – Click OK. The *Signed Certificates* window is displayed.
     – Add the server certificate information:
       - Select ADD. The *Add CA's Certificate from a File* window is displayed.
       - Select data type 'Binary DER data'.

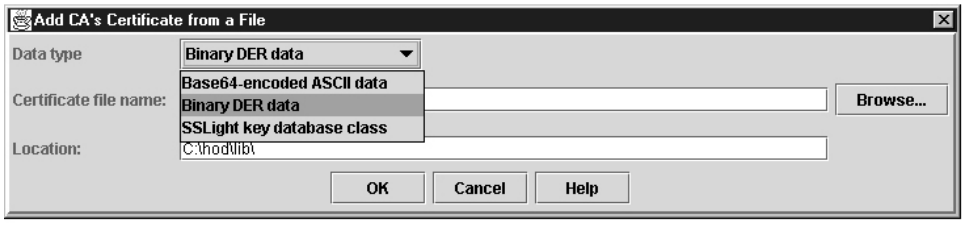


Figure 93. Add CA's Certificate From a File

– Specify the path and name of the binary certificate file that was FTPed from the server or the file extracted using the HOD client window above. Click on the OK button to complete the add.

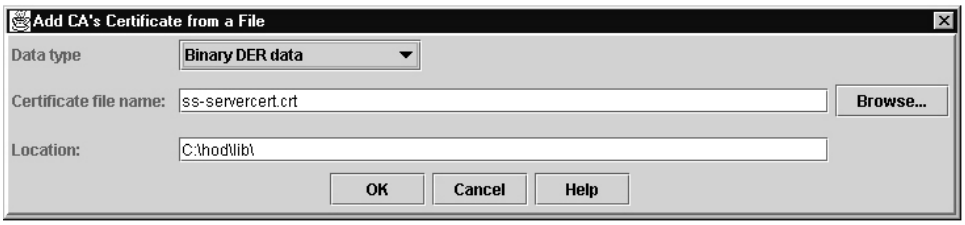


Figure 94. Add CA's Certificate From a File — continued

– Close the CustomizedCAs.class after completing the ADD.

3. Restart HOD to pick up the updated CustomizedCAs.class.

Appendix C. Express Logon Feature (ELF)

Users accessing SNA applications using TN3270 clients such as Host On-Demand (HOD) are generally required to know the user ID and password for the application they want to access. The ID-and-password authentication process creates several potential problems. For example, users may forget their IDs and passwords. If they do forget, the passwords must be reset by a system administrator, a time-consuming process. On the other hand, writing down the IDs and passwords or sharing them with someone else creates a security risk, especially since passwords are usually valid for relatively long periods of time.

IBM's solution to these problems is the Express Logon Feature (ELF), a process which allows a user on a workstation with a TN3270 client and an X.509 certificate to log on to a SNA application without entering an ID or password. The Express Logon Feature is supported on two-tier and three-tier network designs. The two-tier design utilizes the z/OS TN3270 Telnet server. The three-tier design utilizes a middle-tier TN3270 server and a Digital Certificate Access Server (DCAS).

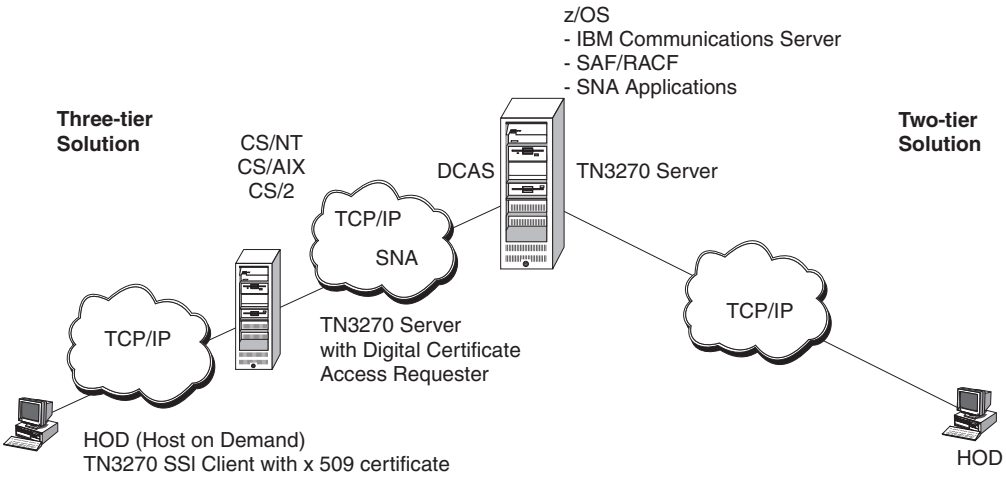


Figure 95. Express Logon network

Both network designs require a TN3270 client workstation that supports Secure Sockets Layer (SSL) connections with client authentication and an X.509 certificate. Using RACF services in z/OS, the client certificate must be associated with a valid user ID. The only client-side product that supports the Express Logon Feature is the IBM WebSphere Host On-Demand V5.0.

The two-tier design requires the z/OS TN3270 Telnet server with SSL, client authentication, and Express Logon functions turned on. Refer to "Express Logon Feature (ELF)" on page 361 for server setup information.

The three-tier design requires a middle-tier TN3270 Telnet server and a Digital Certificate Access Server (DCAS). A middle-tier TN3270 server, so called because it does not reside on the host, but rather between the TN3270 workstation client and the host. This server includes a Digital Certificate Access Requester (DCAR). The middle-tier IBM TN3270 servers supporting Express Logon are:

- CS2 6.1
- CS/NT 6.1.1 PTF
- CS/AIX 6.0.0.1 PTF

Note: The term *DCAR* is used to describe the part of the TN3270 middle-tier server that supports the Express Logon Feature and communicates as a client with the DCAS. It is not separate from the TN3270 middle-tier server. The term *DCAR* might not be used in other documents that describe ELF but has been used here to simplify the description of this function.

A Digital Certificate Access Server (DCAS) resides on the host. DCAS uses RACF services to obtain a user ID.

The host also provides RACF Secured Signon services, which the DCAS or the MVS host Telnet server use to generate a *PassTicket*. A PassTicket is a RACF token similar to a password except that it is valid only for ten minutes.

In a typical scenario, a Host On-Demand (HOD) client wants to log on to a TSO application on the host.

- In the two-tier design, the user starts an SSL connection with level 2 client authentication which passes the client certificate to the MVS host TN3270 server. The MVS host TN3270 server uses RACF Secured Signon services to obtain a user ID and PassTicket.
- In the three-tier design, the user starts the TN3270 connection to the middle-tier server. The DCAS's client is the middle-tier TN3270 server or DCAR, which attempts to log on to an SNA application for the workstation client. The DCAS receives a digital certificate from the DCAR and returns a user ID and PassTicket. SSL communication is used between the DCAS and the DCAR. The server recognizes that the client wants the Express Logon function and invokes the DCAR, which opens an SSL connection with client authentication and passes the workstation's certificate and application name to the DCAS on the host. The DCAS uses RACF Secured Signon services to obtain a user ID and PassTicket, which the DCAS returns to the DCAR. The DCAR passes this information back to the TN3270 server.

In both cases the ELF-enabled client and server now have enough information to complete the logon to TSO. This occurs without the user ever having to enter a user ID or password.

Note: You can use RACF or any other SAF-compliant security product that supports PassTickets with Express Logon. RACF APAR OW44393 is required when using the following:

- TSO with Generic Resources and PTKDATA Class profiles.
- Applications with shared user IDs that could access the application simultaneously. RACF requires the PTKTDATA profile to specify APPLDATA('NO REPLAY PROTECTION.')

For a final list of PTFs, refer to the HOD README file.

Configuring RACF services for Express Logon

At a minimum, you must register all workstation client certificates with RACF using the RACDCERT command. This associates the certificates with the IDs of users attempting to log on. In the two-tier solution, the certificate is passed from the client to the TN3270 server. In the three-tier solution, the certificate is passed from the client to the middle-tier TN3270 server, then to the DCAR, and then to the DCAS.

You must also create a RACF PTKTDATA profile for each application ID the end user is attempting to access. The PTKTDATA profile allows the DCAS or z/OS

TN3270 server to obtain a PassTicket and user ID for the application. In the three-tier solution, the DCAS must pass the passticket and user ID back to the DCAR. For HOD V5, the application ID part of the profile name must be the same as that configured in the HOD V5 Express Logon Application ID popup window. In most cases, the application name with which the user logs on will match the application ID portion of the RACF PTKTDATA class profile. However, for TSO and some other applications, the names and IDs may not match:

- If VTAM generic resources are used for TSO, define the application name portion on the RACF profile using the TCASGNAM defined in the TSOKEYxx, SYS1.PARMLIB member.
- If VTAM generic resources are not used, define the application name on the RACF profile as TSO.
- When configuring for TSO application logon, use the format TSO<SID> in the PassTicket profile, where SID is the SMF system ID defined in the SMFPRMxx member of SYS1.PARMLIB. (For example, if the SID is 3390, you would type TSO3390 in the profile.) For details, refer to the *z/OS Security Server RACF Security Administrator's Guide*.

For applications that allow shared user IDs (multiple users request access to the application simultaneously with the same user ID), you must specify the APPLDATA('NO REPLAY PROTECTION') option on the RDEFINE command in the PTKTDATA profile. This bypasses the default RACF protection against replay of PassTickets.

Configuring the Express Logon components

The following describes, in general terms, how to set up and configure currently supported Express Logon components:

- HOD V5 TN3270 client
- z/OS TN3270 server
- Middle-tier TN3270 server (CS/2 V6.1, CS/NT 6.1.1, and CS/AIX 6.0.0.1)
- DCAS

For details on configuring each product, refer to the appropriate documentation for that product.

Configuring the HOD V5 TN3270 client

To setup and configure the HOD V5 client, follow these steps:

1. For each application to which the end user will log on, create a macro to record the logon screen of the application.

   The end user user plays this macro when displaying the logon screen on the workstation.
2. Enter the application ID in the application ID popup window.

   The application ID must be the same name specified on the z/OS for the application ID portion of the PTKTDATA profile. The ID in the profile in most cases must be the same as the application name.
3. Use the HOD key-management utility to:
   a. Create a key database (keyring).
   b. Create a certificate request or generate a self-signed certificate and associate the certificate with the keyring.
   c. Use FTP to transmit the middle-tier TN3270 certificate to the workstation and store the server certificate in the key database of the client.

d. Use FTP to transmit the TN3270 HOD client certificate to the middle-tier server and store in the SSL key database of the server.

4. Use FTP to transmit the workstation certificate to an MVS data set on the z/OS host. Use the RACF Certificate Services RACDCERT command to associate the certificate with a valid user ID.

Configuring the z/OS TN3270 server

Express Logon Feature requires SSL with level 2 client authentication functionality at the server. Once that level of security is working, specify the EXPRESSLOGON parameter statement to enable ELF in the z/OS TN3270 server.

Configuring the middle-tier TN3270 server (CS/2 example)

The middle-tier server is a TN3270 server, such as CS/2 V6.1, that communicates with the HOD V5 client using an SSL connection with client authentication. The middle-tier server DCAR also communicates with the DCAS on the host. The DCAS and DCAR communicate over a TCP/IP connection using SSL with client authentication.

To configure the TN3270 server, follow these steps:

1. Configure the NDF file for the Express Logon function and communication with the DCAS using the following command:

```
DEFINE_EXPRESS_LOGON_SUPPORT
        ENABLED(YES)
        DCAS_ID(9.25.55.182)
        DCAS_ID_TYPE(IP_ADDRESS)
        DCAS_PORT(8990)
```

2. Use the local key management utility to store the workstation client certificate and the DCAS certificate in the local keyring:
   a. Create a key database file.
   b. Create a certificate request or generate a self-signed certificate and associate the certificate with the keyring.
   c. Store the workstation client certificate and the DCAS certificate in the keyring of the server.
3. Use FTP to transmit the DCAR certificate to the z/OS host and use gskkyman or RACF Certificate Services to store the DCAR certificate in the DCAS keyring.

Configuring the Digital Certificate Access Server (DCAS)

Setting up DCAS

The DCAS must run from a user ID defined with a UID=0 and from an APF authorized library. The shipped executable resides in /usr/lpp/tcpip/sbin with the sticky bit on. The DCAS uses the z/OS SSL product, shipped with the z/OS base element. The SSL library *hlq*.SGSKLOAD must be in the run-time STEPLIB.

The default port used by the DCAS is port 8990. If you want to ensure that no other application uses the same port, use the following statement in the TCPIP profile data set:

```
PORT
    8990 TCP DCAS
```

If you choose to run the DCAS from the z/OS UNIX shell, use the following statement:

```
PORT
   8990 TCP OMVS
```

It is recommended that you use port access controls. For details on access controls, refer to the *z/OS Communications Server: IP Migration*.

Create a configuration file for the DCAS taking into account the following:

- Define the port DCAS will listen on or use the default 8990.
- Define the KEYRING or SAFKEYRING for SSL communication.
- Decide the type of CLIENTAUTH needed.

For details on configuring the DCAS, see *z/OS Communications Server: IP Configuration Reference*.

Define a user ID as superuser to OMVS services

The server requires the user ID from which it is started be defined to use OMVS services as a superuser. The OMVS(UID(0)) could be put on the ADDUSER command. However, if the user ID already exists the ADDUSER will fail and the user ID will not be altered to superuser. ALTUSER will set the user ID to superuser whether the user ID existed before or was just created by the ADDUSER command.

```
ADDUSER DCAS
ALTUSER DCAS   DFLTGRP(OMVS)  OMVS(UID(0)HOME('/'))
```

Give the user ID access to operator commands

If the server is started from an MVS procedure, it is required that the user ID have access to the appropriate server resources in the OPERCMDS class. Use the following commands to provide access:

```
RDEFINE OPERCMDS(MVS.SERVMGR.DCAS) UACC(NONE)
PERMIT MVS.SERVMGR.DCAS  CLASS(OPERCMDS) ACCESS(CONTROL) ID(DCAS)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Provide a RACF definition for MVS startup

If the server is started as an MVS procedure, use the following RACF definitions to define the server to RACF:

```
RDEFINE STARTED  DCAS.* STDATA(USER(DCAS))
SETROPTS RACLIST(STARTED)  REFRESH
```

Starting, stopping, and toggling DCAS

DCAS can be started as either a generic server without stack affinity or as a server with affinity to a specific TCP/IP stack.

1. Start the DCAS.

   You can start the DCAS from the z/OS UNIX shell or with an MVS started procedure using optional parameters for debugging, logging, and specifying the configuration file. Refer to the *z/OS Communications Server: IP Configuration Reference* for information on parameters used to start DCAS.

You can start the DCAS automatically when the TCP/IP address space is started or from the z/OS UNIX shell:

- To start the DCAS automatically when the TCP/IP address space is started, specify DCAS on the AUTOLOG statement in the TCPIP profile data set as shown in the following example:

  ```
  AUTOLOG
    DCAS
  ENDAUTOLOG
  ```

  A sample start procedure for the DCAS is provided in *hlq*.SEZAINST(EZADCASP).

- To start the DCAS from the UNIX shell, use the **dcas** command with optional parameters. (Refer to *z/OS Communications Server: IP Configuration Reference*) Note that the DCAS must run as a background job.

When DCAS is started, it stores its process ID (pid) in a Hierarchical File System (HFS) file. The file name under which it is stored depends upon how you configure DCAS:

- If you configure the DCAS with TCP/IP stack affinity, the pid file is named /tmp/dcas.tcpipname.pid where tcpipname is the name of the TCP/IP stack for which DCAS has affinity.
- If you configure the DCAS without stack affinity, the process ID file is named /tmp/dcas.INET.pid.

You can stop the DCAS from the UNIX shell or from MVS:

- To stop DCAS from the UNIX shell, use the following command:

```
kill  -s SIGTERM  pid
```

- To stop the DCAS from MVS, use the following command:

```
P DCAS
```

If the DCAS was started without debugging, you can toggle it (turn it on and off):

- To toggle from the z/OS UNIX shell, use the following command:

```
kill -s SIGHUP pid
```

- To toggle for an MVS started procedure, use the MODIFY command:

```
F DCAS,x
```

where *x* is any valid character.

DCAS and system SSL

The DCAS and the DCAR use SSL V3 to communicate. The SSL protocol begins with a handshake. Then, the DCAR authenticates the DCAS and vice versa. At this time, the DCAS and the DCAR also agree on how to encrypt and decrypt the data.

You can specify the cipher level used for encryption and decryption for each connection at the time DCAS is configured, using the V3CIPHER configuration keyword. Alternatively, you can set the cipher level dynamically when DCAS starts, based on the level of cipher installed on the system. To set the cipher level dynamically, do not specify the V3CIPHER keyword.

Refer to Appendix B, “TLS/SSL security” on page 721 for information on creating and managing keyrings and certificates using either:

- The gskkyman tool
- The RACDCERT command

Authenticating the DCAS and the DCAR: The type of security and authentication required will determine the way certificates are created and managed. The DCAS, in conjunction with SSL and RACF, supports several levels of authentication.

Authenticating the DCAS: DCAS authentication is always performed by the DCAR. Authentication requires that the DCAS has a private key and an associated X.509 digital certificate defined in a keyring.

Authenticating the DCAR: The DCAR is the client that interacts with the DCAS. Authenticating the DCAR involves additional levels of control in which the client

must have a key database with a certificate. Depending on the control level, the certificate is authenticated by SSL and the DCAS using RACF services.

There are three levels of client authentication from which to choose. Refer to Appendix B, “TLS/SSL security” on page 721 for details.

To configure DCAS for level 1 authentication, specify the CLIENTAUTH LOCAL1 keyword and value in the DCAS configuration file. Use the KEYRING or the SAFKEYRING keywords in the DCAS configuration file to specify the keyring used by the DCAS.

To configure DCAS for level 2 authentication, specify the CLIENTAUTH LOCAL2 keyword and value in the DCAS configuration file.

If CLIENTAUTH LOCAL2 is coded in the DCAS configuration file, at a minimum, you must use RACF to associate the DCAR certificate with a valid user ID. You can do this using the RACDCERT ADD command. The user ID could be the one associated with the DCAS itself or it could be any valid user ID. If you want additional checking, you must activate the SERVAUTH class and define an EZA.DCAS.cvtsysname profile with the user ID associated with the DCAR certificate to access the profile.

Appendix D. Using HCD

The information in this chapter shows examples of panels that are used to define IQD channels and devices for z/OS Communications Server using HCD.

1. Select processors

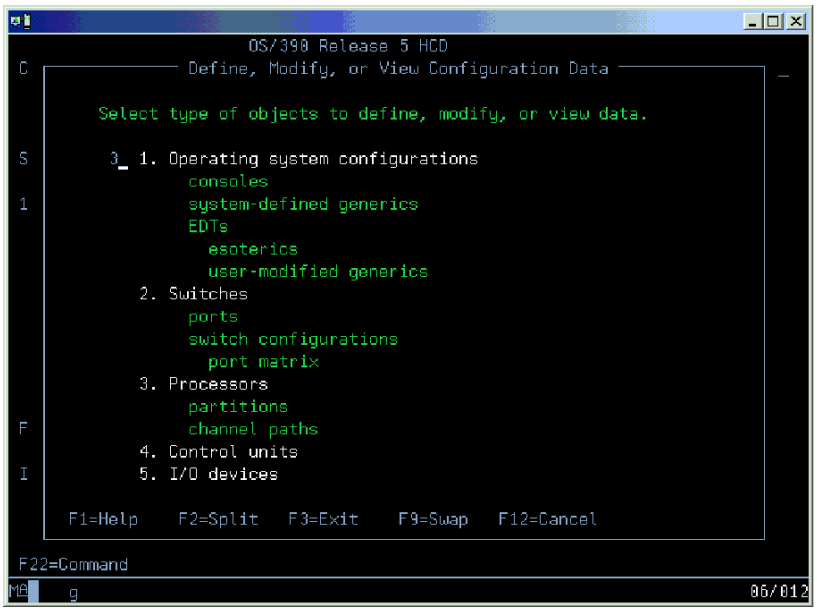


Figure 96. Select processors

2. Select 'S' Work with attached channel paths

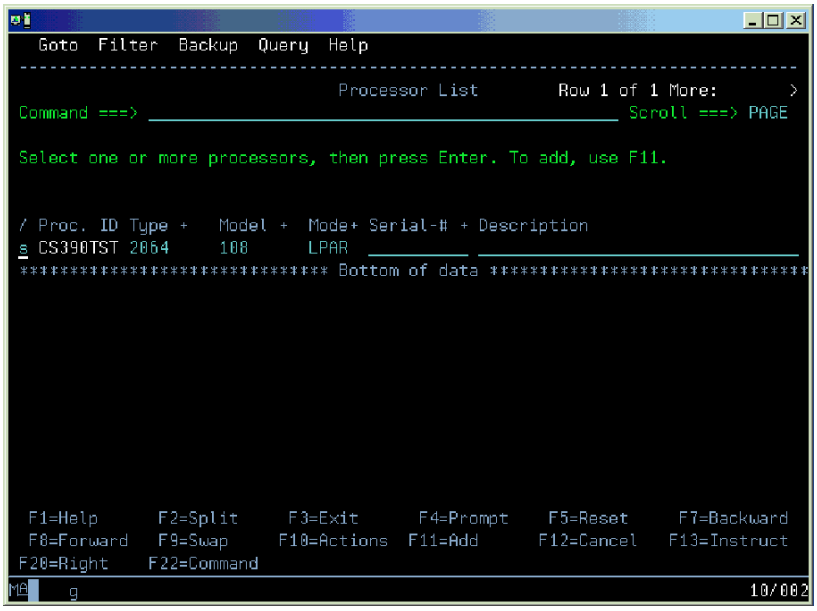


Figure 97. Work with attached channel paths

3. On the Channel Path List enter the Add command (or press F11) to initiate the Define Channel Path dialog.

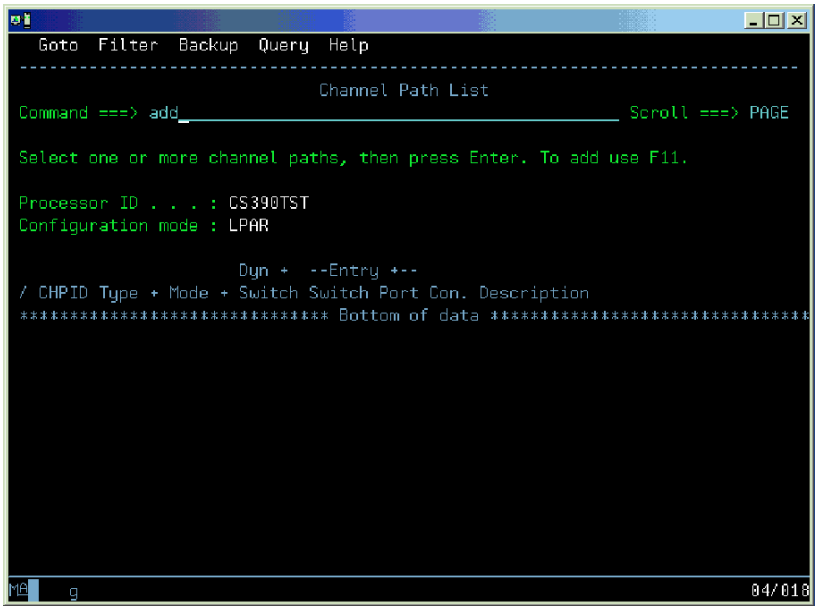


Figure 98. Initiate the Define Channel Path dialog

4. Fill in the Add Channel Path panel, then press Enter (Select SHR for Operation mode to share IQD Chpids across LPARs).

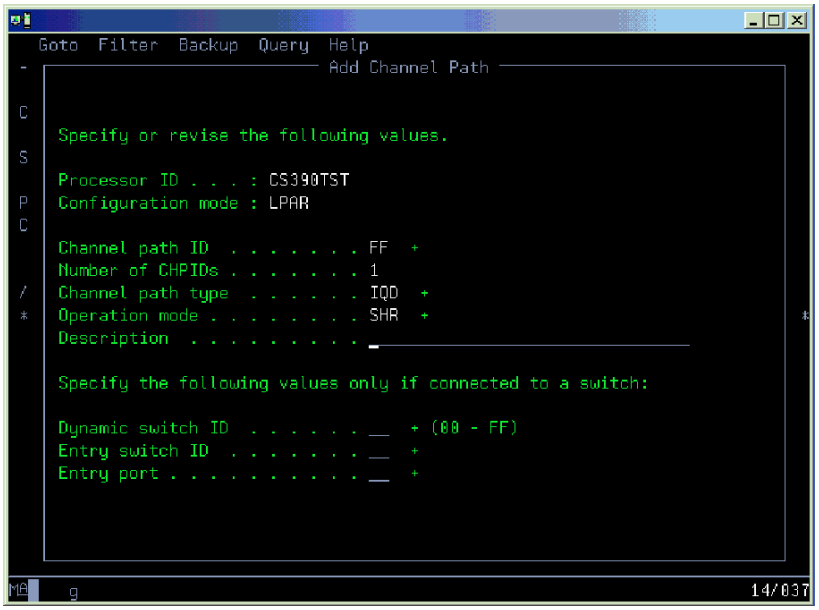


Figure 99. Add channel path

5. For an IQD channel path, the Specify Maximum Frame Size panel pops up with the default value of 16 KB.

Figure 100. Specify Maximum Frame Size

Or change the frame size to desired size:

Table 26. Frame size specification

Maximum Frame Size	TCP/IP MTU size
16K	8K
24K	16K
40K	32K
64K	56K

6. Define the channel path access list that each LPAR should have access to.

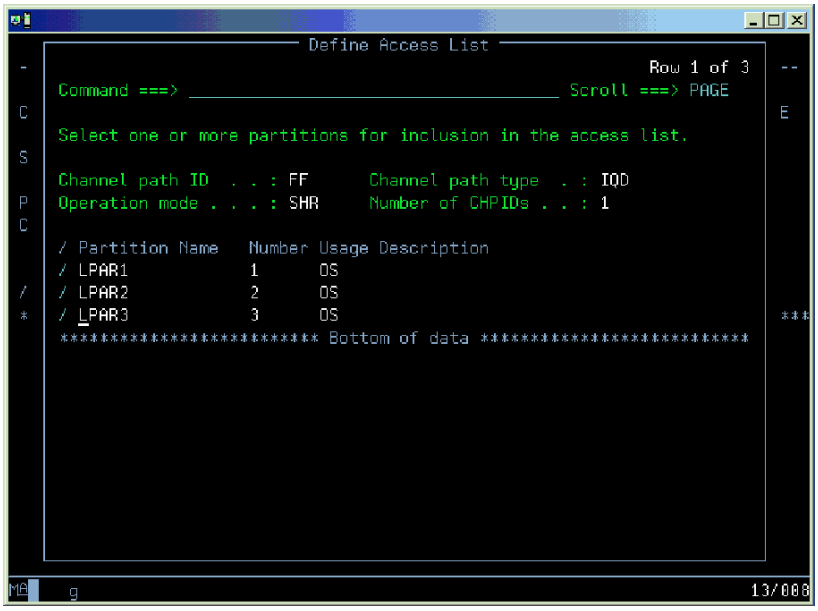


Figure 101. Define the channel path access list

7. Having pressed Enter, the Channel Path List is redisplayed with channel path number FF defined.

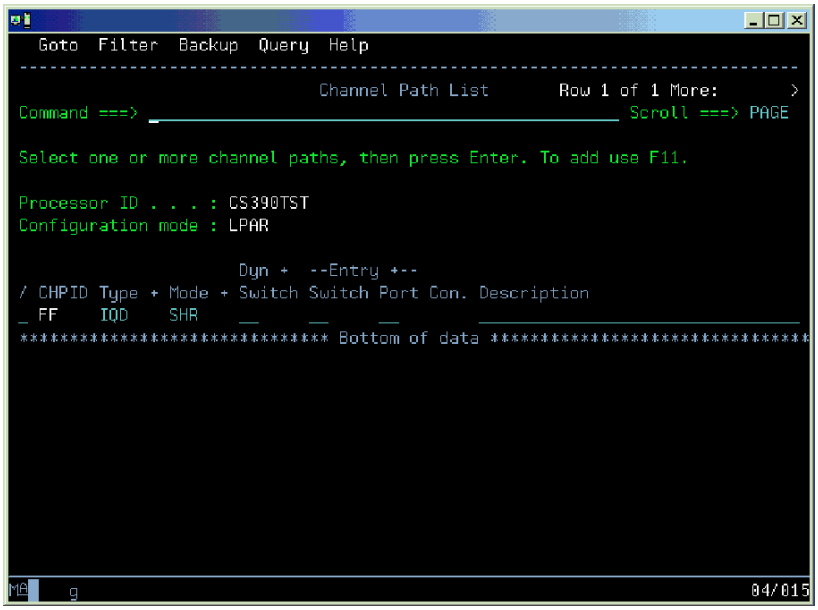


Figure 102. Channel path number FF defined

8. As the next step, add the control unit(s) to the IQD channel path. Select the defined channel path with action ″Work with attached control units″ (action code 'S').

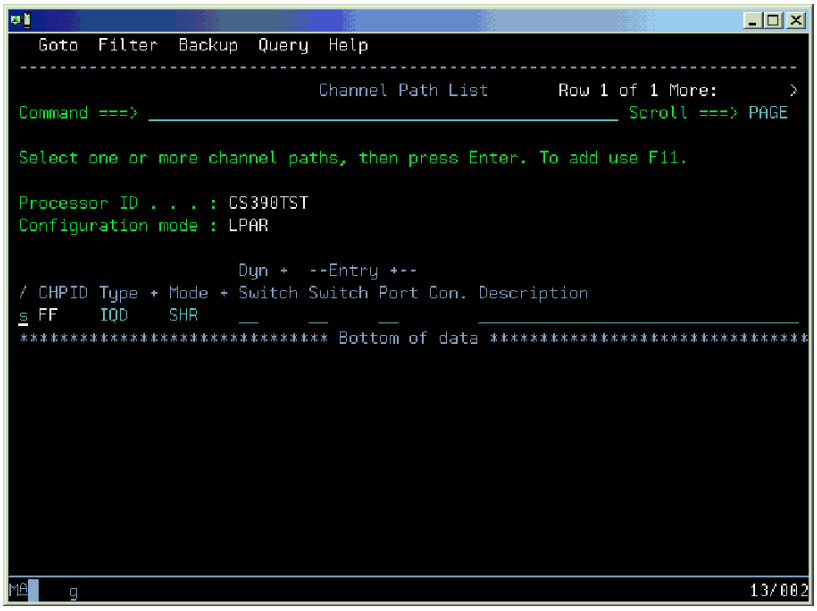


Figure 103. Work with attached control units

9. An empty control unit list is displayed. Enter the 'Add' command or F11.

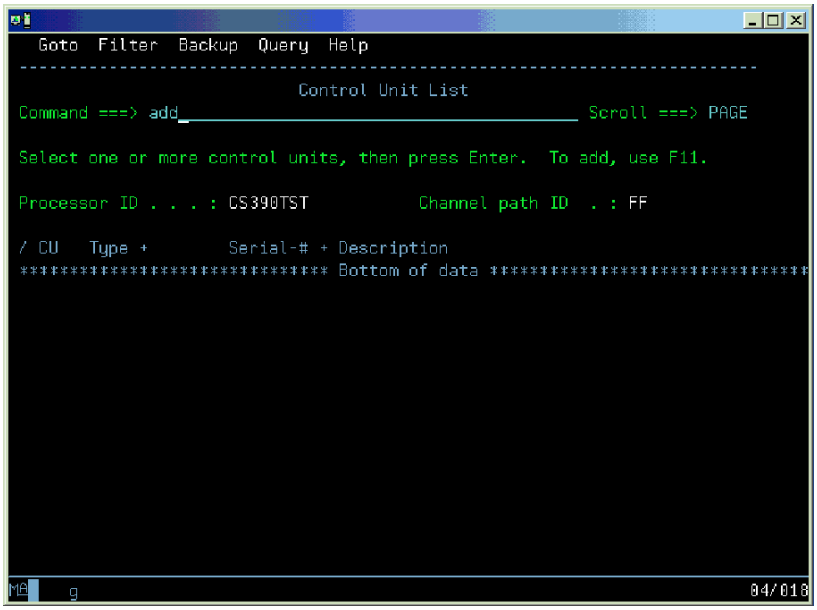


Figure 104. Add the control unit(s)

10. Define a control unit of type 'IQD' for channel path FF.

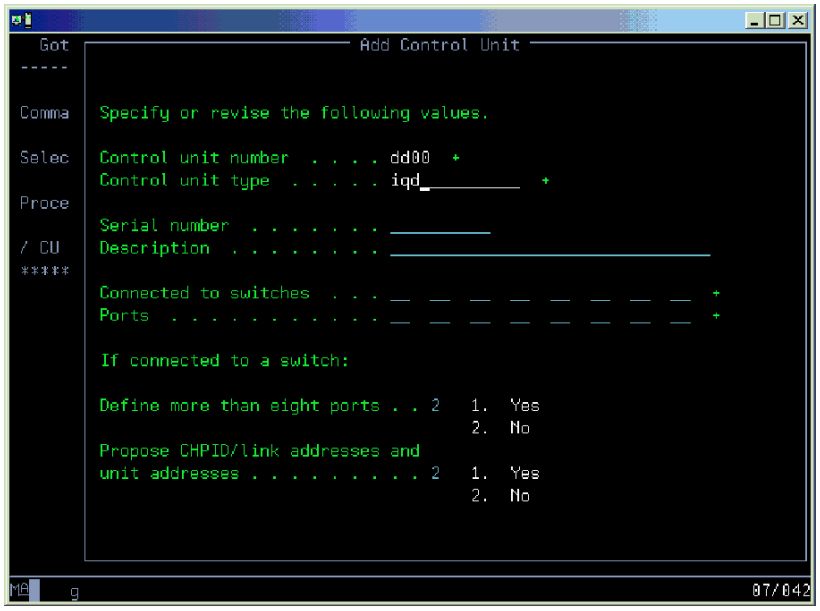


Figure 105. Define a control unit

11. Define it to the processor:

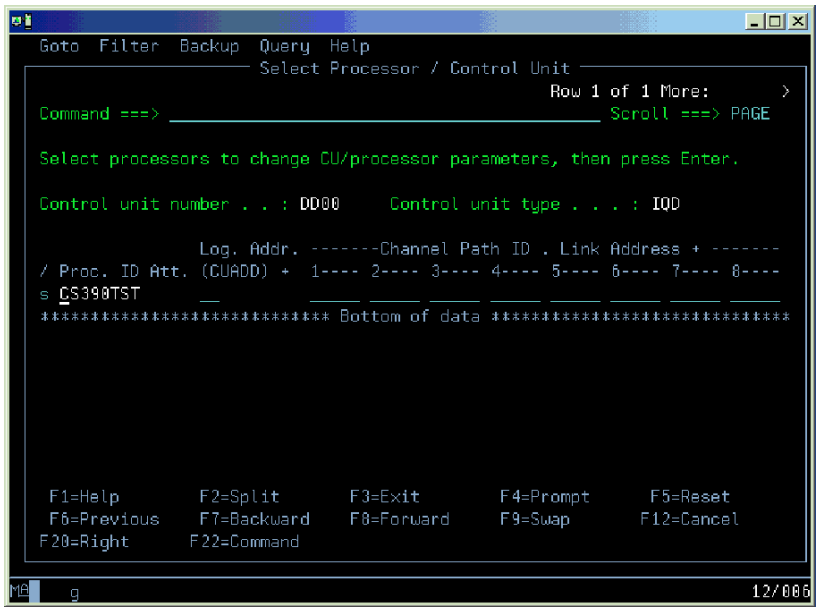


Figure 106. Define it to the processor

12. The processor settings are already preset. Pressing Enter, returns to the Select Processor/Control unit panel. Pressing Enter again, returns to the Control Unit List panel which shows the currently defined control unit.

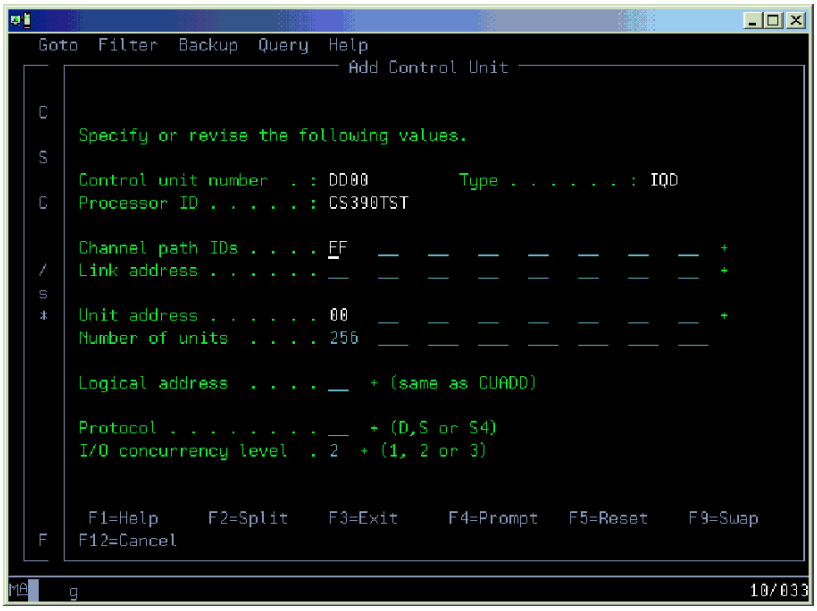


Figure 107. Currently defined control unit

13. Next, define the devices. Successively, select a control unit and perform action ″Work with attached Devices″.

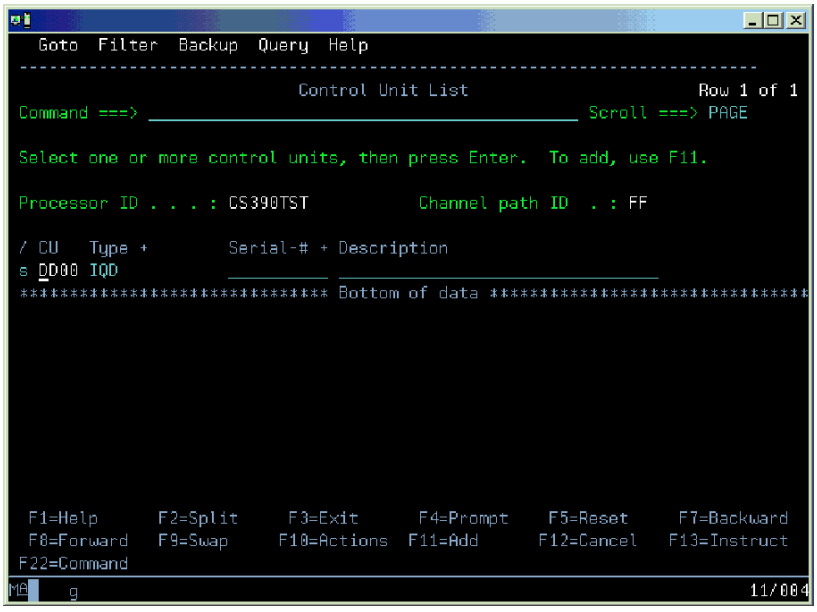


Figure 108. Define the devices

14. This leads to an empty device list.

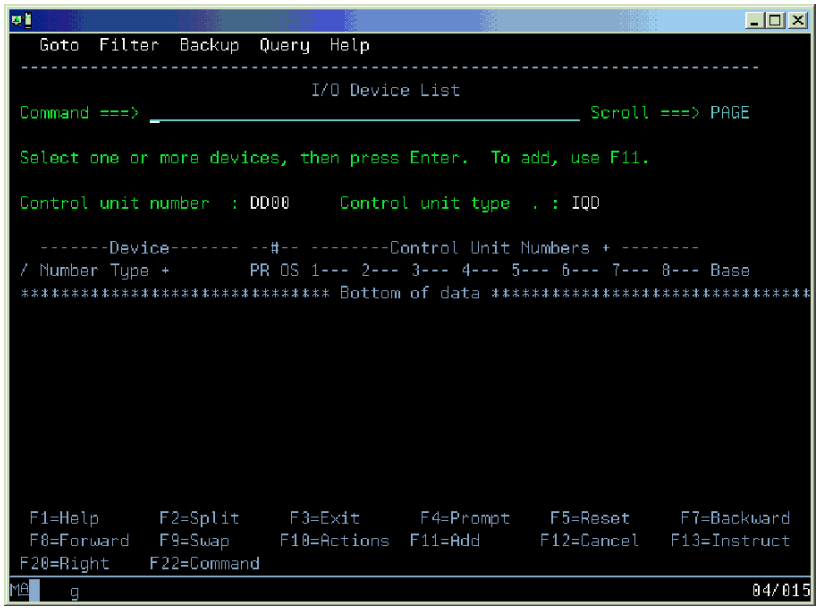


Figure 109. Empty device list

15. Perform the Add action to define the devices for the control unit selected in the previous step.

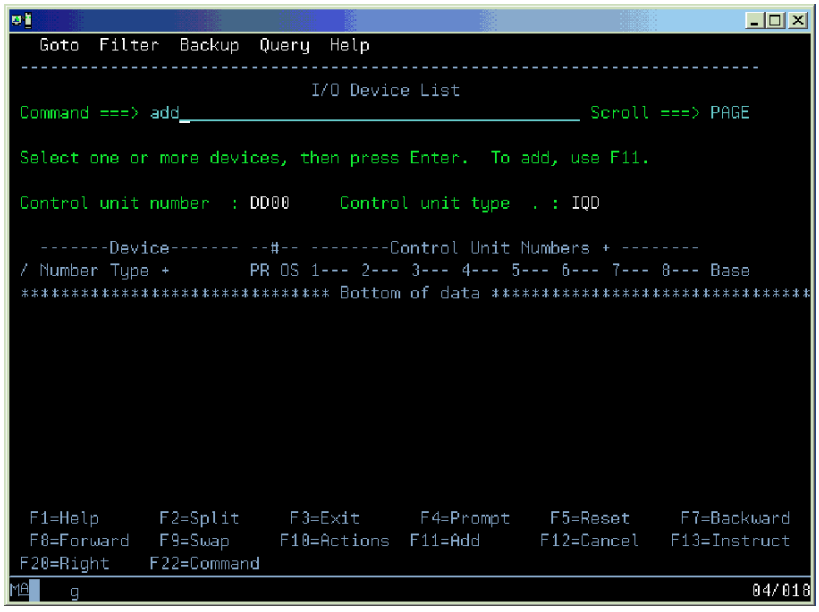


Figure 110. Define the devices for the control unit

16. Add devices of type IQD to the selected control unit.

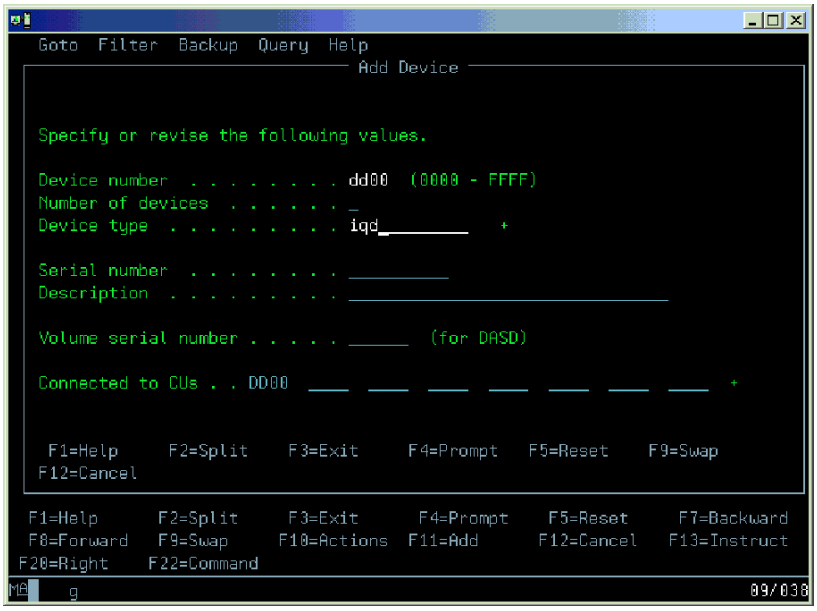


Figure 111. Add devices of type IQD

17. If the number of devices has been left unspecified (as in this example), HCD defines 10 devices.

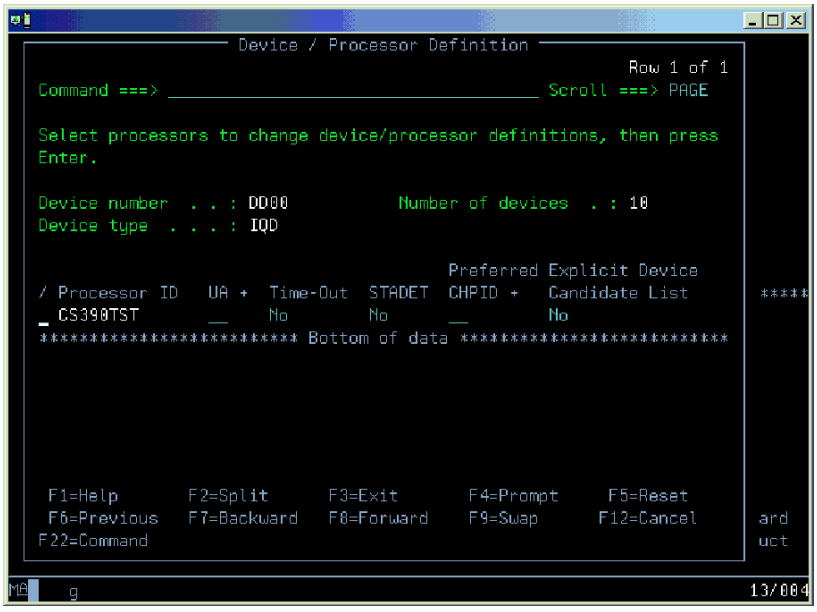


Figure 112. Define number of devices

18. Hit enter and the next panel displayed will be:

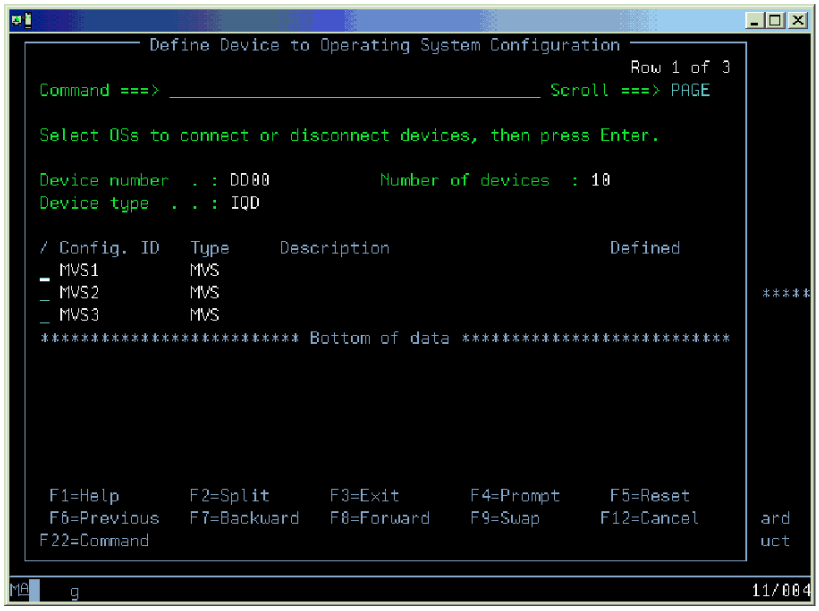


Figure 113. Define device to operating system

19. Next, define the devices to the operating system by selecting an 'S' on each system you want to have them defined on.

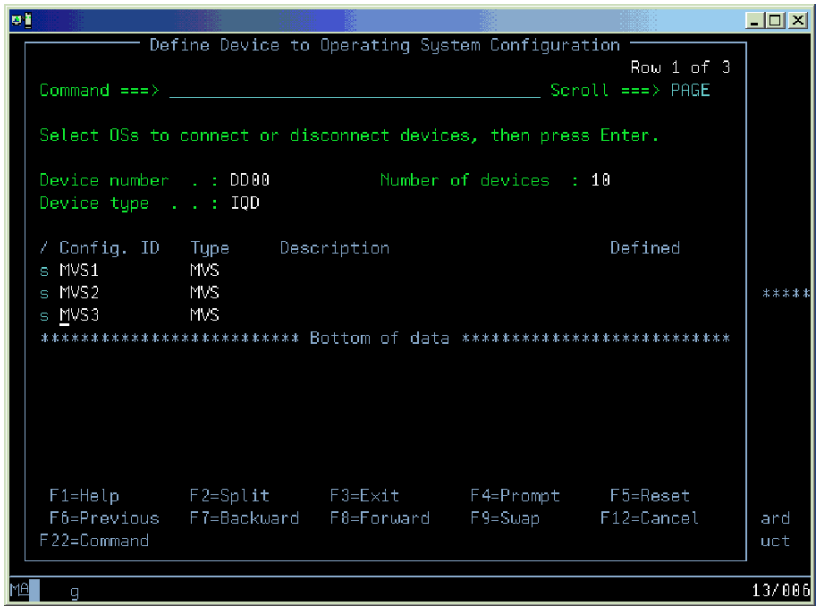


Figure 114. Select systems

20. The device parameters are shown with default values. Press Enter, to complete the definition for each system.

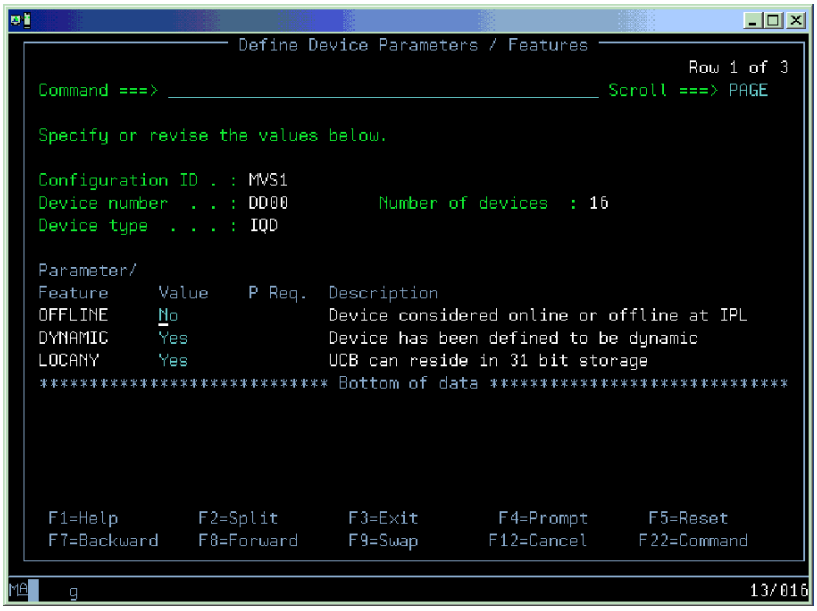


Figure 115. Complete the definition

21. Press Enter until you return to the I/O device list panel, the definition for channel path FF is complete.

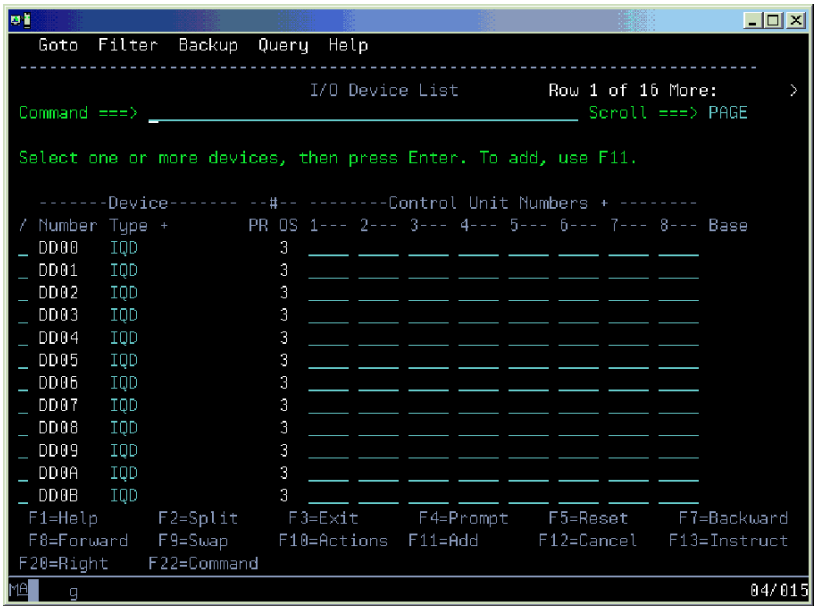


Figure 116. Definition completed

The sample IOCP input for this example would be:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
         CHPID PATH=(FF),SHARED,                                        *
               PARTITION=((LPAR1,LPAR2,LPAR3),(LPAR1,LPAR2,LPAR3)),     *
               TYPE=IQD,OS=00
         CNTLUNIT CUNUMBR=DD00,PATH=(FF),UNIT=IQD
         IODEVICE ADDRESS=(DD00,010),CUNUMBR=(DD00),UNIT=IQD
```

Appendix E. Configuring the OROUTED server

Before you configure...

Read and understand Chapter 1, “Configuration overview” on page 3. It covers important information about data set naming and search sequences.

This appendix describes how to configure the OROUTED server. It explains OROUTED’s use of the Routing Information Protocol to help you decide if this server is suitable for your network.

Notes:

1. OMPROUTE, which was introduced in eNetwork Communications Server for OS/390 V2R6 and enhanced in V2R7, is the recommended routing daemon. OROUTED will eventually be removed; you will receive ample formal notice of this change.
2. OROUTED does not support zero subnets.
3. OROUTED does not support equal-cost multipath routes to a destination network or host. However, OROUTED can be used in conjunction with statically defined equal-cost multipath routes in the GATEWAY statement of the TCP/IP profile.

Understanding OROUTED

The route daemon is a server that implements the Routing Information Protocols (RIP) described in RFC 1058 (RIP version 1) and in RFC 1723 (RIP version 2). It provides an alternative to the static TCP/IP gateway definitions. When configured properly the MVS host running with OROUTED becomes an active RIP router in a TCP/IP network. The OROUTED server dynamically creates and maintainsthe network routing tables using RIP. RIP allows gateways and routers to periodically broadcast their routing tables to adjacent nodes. This enables the OROUTED server to update the host routing table. For example, the OROUTED server can determine if a new route has been created, if a route is temporarily unavailable, or if a more efficient route exists. OROUTED has the following characteristics:

- Deletion of all RIP routes at startup. The -del start parameter might be used to delete all dynamic routes from the routing table upon initialization of OROUTED.
- OROUTED is a z/OS UNIX application. It requires the Hierarchical File System (HFS) to run.
- OROUTED can be started from an MVS procedure or from the z/OS shell command line.
- OROUTED uses a standard message catalog. The message catalog must be in the HFS. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.
- All messages and trace information is sent to the syslogd, except for output from the *-d* and *-dp* parameters, which is sent to STDOUT.
- A default mode of operation is for the program to close STDIN, STDOUT and STDERR. This allows for userids to cleanly exit the z/OS shell after starting the program in the background. As a consequence, the program printf statements are also disabled. The parameter ″-ep″ enables printf’s. If this parameter is specified, the program should not be run in the background, because the userid will not be able to exit the shell until the background job has been killed.
- OROUTED and OMPROUTE cannot run on the same stack concurrently.

- OROUTED needs to be started by a RACF-authorized user ID.

Routing Information Protocol (RIP)

The Routing Information Protocol (RIP) is an Interior Gateway Protocol (IGP) designed to manage a relatively small network. IGPs are used to manage the routing information of a single autonomous system, or a single piece of the TCP/IP network. RIP has many limitations and is not suited for every TCP/IP environment. Before installing the OROUTED server, read RFCs 1058 and 1723 to decide if RIP can be used to manage the routing tables of your network. See Appendix F, "Related protocol specifications (RFCs)" on page 797 for more information about RFC 1058 and RFC 1723.

RIP uses the number of hops, or *hop count*, to determine the best possible route to a host or network. The term *hop count* is also referred to as the *metric*. A gateway is defined as zero hops from directly connected networks, one hop from networks that can be reached through one gateway, and so on. In RIP, a hop count of 16 means infinity, or that the destination cannot be reached. This limits the longest path in the network that can be managed by RIP to 15 gateways.

The OROUTED server propagates routing information to the neighboring gateway's on gateway's directly connected networks every 30 seconds. The server receives updates from neighboring gateways periodically and uses this information to update the routing tables. If an update has not been received from a gateway in 180 seconds (3 minutes), OROUTED assumes the gateway is down and sets all the routes through that gateway to a metric of 16 (infinity). If an update has not been received from a gateway in another 120 seconds (2 minutes), OROUTED deletes all of the routes through that gateway.

During the intervals specified by the *interface.scan.interval* and *interface.poll.interval* values on the OPTIONS statement, OROUTED checks to determine if a local interface is up or down by scanning the TCP/IP interface tables. It also checks to see if an interface has been added or reactivated.

For networks that are not point-to-point, such as Token-Ring and Ethernet, OROUTED receives its own copy of broadcasted or multicasted RIP packets over the interfaces, provided that the interfaces are active. Other networks, such as point-to-point, can be managed by OROUTED as long as there are RIP services managing the other end of point-to-point link(s). If the destination addresses are defined for these point-to-point networks, the RIP packets are unicasted. Otherwise, the RIP packets are broadcasted or multicasted. In networks where there are adjacent routers or hosts not running RIP services, OROUTED will not be receiving updates over the links and eventually will delete all of the routes to these networks. To prevent the routes to these networks not running RIP services from being deleted, passive routes may be coded in a OROUTED gateways data set. For more information, see "OROUTED gateways" on page 773 and "OROUTED parameters" on page 786

Because OROUTED requires link-level broadcasting or multicasting to send routing updates, it requires routers that do not support link-level broadcasting or multicasting (for example, HYPERchannel and ATM) to be active gateways. For more information on how to manage networks without link-level broadcasting or multicasting support, see "Active gateways" on page 773 and "Configuring an active gateway" on page 791. By configuring active gateways, the RIP packets are unicasted to the neighboring gateways.

RIP Version 2

RIP Version 2 is an extension of RIP Version 1 and provides the following features:

Route Tags to provide EGP-RIP and BGP-RIP interactions
The route tags are used to separate ″internal″ RIP routes (routes for networks within the RIP routing domain) from ″external″ RIP routes, which may have been imported from an EGP or another IGP. OROUTED will not generate route tags, but will preserve them in received routes and readvertise them when necessary.

Variable subnetting support
Variable-length subnet masks are being included in routing information so that dynamically-added routes to destinations outside subnetworks or networks can be reachable.

Immediate next hop for shorter paths
Next hop IP addresses, whenever applicable, are being included in the routing information. Its purpose is to eliminate packets being routed through extra hops in the network. OROUTED will not generate immediate next hops, but will preserve them if they are included in the RIP packets.

Multicasting for RIPv2 packets to reduce load on hosts
An IP multicast address 224.0.0.9, reserved for RIPv2 packets, is used to reduce unnecessary load on hosts which are not listening to RIPv2 messages. RIPv2 multicasting is dependent on interfaces that are multicast-capable. By default, RIPv1 packets will be broadcast for interfaces that are not multicast-capable.

Authentication of RIPv2 packets for routing update security
Authentication keys, consisting of passwords, can be configured to be included in the outgoing RIPv2 packets for authentication by adjacent routers as a routing update security protection. Likewise, incoming RIPv2 packets are checked against local authentication keys. Any outgoing or incoming RIPv1 packets are not authenticated. For maximum security, configure OROUTED such that it will supply and receive RIPv2 packets only in addition to specification of authentication keys. The authentication keys are configurable on a router-wide or per-interface basis.

Configuration switches for RIPv1 and RIPv2 packets
Configuration switches are provided to selectively control which versions of RIP packets are to be sent or received over network interfaces. The switches should be configured based upon the routing capabilities of the network and are configurable on a router-wide or per-interface basis.

Supernetting support
The supernetting feature is part of Classless InterDomain Routing (CIDR) function. Supernetting provides a way to combine multiple network routes into fewer 'supernet' routes. This means that the number of network routes in the routing tables becomes smaller for advertisements. Supernet routes are received and sent in RIPv2 messages. If local supernet routes are defined for OROUTED, they will be advertised to adjacent routers. Local supernet routes are generated by OROUTED for interfaces with subnet masks that are less than the network class mask in value.

OROUTED miscellaneous features

OROUTED supports the following miscellaneous features:

- Multiple Network Attachments.

Multiple attachments and IP addresses on the same network are supported, providing redundant paths to other hosts or routers on directly-attached networks.

- Virtual IP Addressing (VIPA).

  If virtual IP addresses are configured on the z/OS server and there are multiple network attachments, OROUTED can be used to advertise the VIPA routes to the network, providing the necessary routing information to the adjacent routers or hosts running RIP services. Using the VIPA routes, the adjacent routers or hosts will be able to reach the VIPA addresses as the destinations and to route around failures for fault tolerance support. For more information, see Chapter 5, “Virtual IP Addressing” on page 209.

RIP input/output filters

The RIP input/output filters provide routing table manipulation and routing control. The filters are provided by OROUTED and consist of:

1. Route Blocking (or NoReceiving)
2. Route Forwarding (Unconditional and Conditional)
3. Route Receiving (Unconditional and Conditional)
4. Route NoForwarding
5. Interface Supply Switch
6. Interface RIP On/Off Switch
7. Default Route Only Supply Switch
8. Virtual Route Only Supply Switch
9. Default and Virtual Routes Only Supply Switch
10. Local (directly-connected) Routes Only Supply Switch
11. Triggered Updates Only Supply Switch
12. Gateway NoReceiving

For more information on these RIP input/output filters, see the OROUTED procedure parameters in “OROUTED parameters” on page 786 and the options statement in “Step 6: Configure the gateways file or data set (optional)” on page 779.

RIP routes

Dynamic routes are any routes created by the following dynamic routing applications:

- OMPROUTE (OSPF/RIP)
- OROUTED (RIP)
- NCPROUTE (RIP)
- ICMP Redirects
- Other (created by customer application)

However, RIP routes are dynamic routes created only by RIP applications, like OROUTED and OMPROUTE.

To help maintain the integrity of the routing table, at initialization, OROUTED attempts to delete all RIP routes from the stack routing table and then adds RIP routes based on BSDROUTINGPARMS and the optional gateways file. The -del start parameter can be used to delete all dynamic routes from the routing table upon initialization of OROUTED.

When OROUTED terminates, RIP routes are not deleted. If you want to remove all RIP routes upon OROUTED termination, used the -kdr MODIFY option.

OROUTED gateways

Passive RIP routes

Passive RIP routes are known by both TCP/IP and OROUTED. Information about passive routes is put in TCP/IP's and OROUTED's routing tables. A passive entry in OROUTED's routing table is used as a placeholder to prevent a route from being propagated and from being overwritten by a competing RIP route. With the exception of directly-connected passive routes, passive routes are not propagated; they are known only by this router. Using passive routes can create routing loops, so they need to be created carefully.

Defining passive routes such as these should be avoided:
A to C is via B.
B to C is via A.

Passive routes should be used when adding routes where the host/net is not running RIP. Passive routes should also be used when adding a default route, since this is the only way to prevent a route from timing out.

External RIP routes

External RIP routes are known by OROUTED, but not by TCP/IP. External routes, such as the External Gateway Protocol (EGP), are managed by other protocols. The OROUTED server needs to know not to interfere with these and not to delete them.

An external entry exists in OROUTED's routing tables as a place holder to prevent a route from being overwritten by a competing RIP route. External routes are not propagated. OROUTED does not manage external routes. Therefore, OROUTED only knows that there is an existing route to host/net and one that is known to TCP/IP.

External routes should be used when the local host is running with some type of non-RIP routing protocol which dynamically changes the TCP/IP routing tables. The foreign host does not need to run any routing protocol, since the only concern is how to route traffic from the local host to the foreign host, and how to prevent multiple routing protocols from interfering with each other.

Active gateways

Active gateways are treated as remote network interfaces. Active gateways are routers which are running RIP, but are reached by a medium which does not allow broadcasting or multicasting and is not point-to-point. OROUTED normally requires that routers be reachable via broadcast for non-point-to-point links or via unicast addresses for point-to-point links. If the interface is neither, then an active gateway entry can add the gateway to OROUTED's interface list. OROUTED will treat the active gateway as a remote network interface. Note that the active gateway must be directly connected.

Active gateways should be used when the foreign router is reachable over a non-broadcast and non-point-to-point network, and is directly connected to the local host.

OROUTED will communicate with active routers by unicast transmissions to the gateway address. Routes are not added to either OROUTED or the TCP/IP routing table immediately. They are added and propagated normally when route advertisements arrive from an active gateway. The sole effect of an active gateway statement is to bypass the requirement for broadcast or multicast communication on non-point-to-point links. Interfaces which are not broadcast-capable or multicast-capable, not point-to-point, and are not active gateways are assumed to be loopback interfaces to the local host. Also, while a route to an active gateway might time out, the interface entry is never removed. If transmissions resume, then the new routes will still be available to the active gateways.

OROUTED gateways summary

Table 27 provides a summary of the OROUTED gateways and their characteristics.

Table 27. OROUTED gateways summary

	Propagated?	Known by TCP/IP?	Known by OROUTED?	Timeout?
Dynamic (1)	Yes	Yes	Yes	Yes
Passive	No (2)	Yes	Yes	No
External	No	No	Yes	No
Active	Yes	Yes	Yes	Yes

Dynamic routing provided by OROUTED.

Notes:

1. Except directly-connected passive routes.
2. Directly-connected passive routes are propagated to other network interfaces for network reachability. A directly-connected passive route is one where the gateway address is one of the local interfaces in the HOME list.

OROUTED configuration process

The steps to configure OROUTED are as follows:

1. Configure the OROUTED profile.
2. Update PORT, BSDROUTINGPARMS, GATEWAY, BEGINROUTES, and IPCONFIG statements in the TCPIP profile.
3. Update the resolver configuration file.
4. Update the OROUTED cataloged procedure (optional).
5. Specify the OROUTED port number in the SERVICES file or data set.
6. Configure the gateways file or data set (optional).

   Note: If a default route is to be defined to a destination gateway or router, configure a default route in this gateways file or data set.
7. If not already started, configure and start syslogd.
8. RACF-authorize user IDs for starting OROUTED.

Step 1: Configure the OROUTED profile

OROUTED supports sending and receiving both RIP version 1 and RIP version 2 packets. A configuration file, the OROUTED profile, determines the mode of operation. The following is the search order used to locate the OROUTED configuration data set or file:

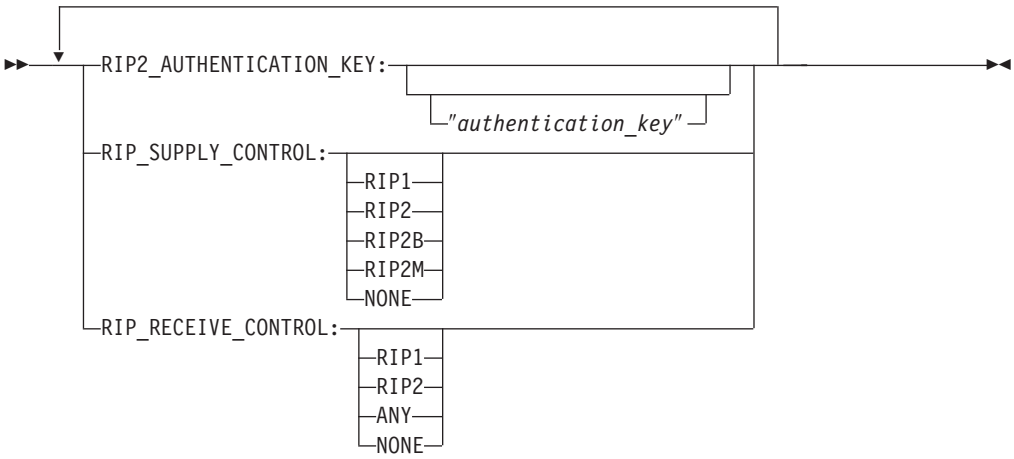
1. If the environment variable ROUTED_PROFILE has been defined, OROUTED uses this value as the name of an MVS data set *(//'mvs.dataset.name')* or HFS file *(/dir/subdir/file.name)* to access the profile.
2. /etc/routed.profile
3. *hlq*.ROUTED.PROFILE

The following are the syntax rules for the OROUTED profile:

- Keywords can be specified in mixed case.
- Blanks and comments are supported in the OROUTED profile. Comments are identified by a semicolon in any column.
- Profile statements may start in any column; however, wrapping to the next record for continuation is not allowed.
- There should be no sequence numbers in the data set or file.

A sample profile is provided in *hlq*.SEZAINST(EZARTPRF).

The following are the options that can be included in the OROUTED profile:



RIP2_AUTHENTICATION_KEY:
A constant. The value that follows is the authentication key.

authentication key
Specifies a plain text password containing up to 16 characters. The authentication key must be enclosed in double quotes. The key is used on a server-wide basis and can contain mixed case and blank characters. The key will be used to authenticate RIP Version 2 packets and be included in the RIP responses for authentication by adjacent routers running RIP Version 2. A null key (either no key is specified, or "" is specified) indicates that authentication is disabled. For maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL to RIP2. This will discard RIP1 and unauthenticated RIP2 packets.

RIP_SUPPLY_CONTROL:
A constant. Specifies that the keyword following is to be used as the RIP supply control for all interfaces. Possible supply controls are as follows:

RIP1 Unicast/Broadcast RIP Version 1 packets (Default)

RIP2 Unicast/Multicast RIP Version 2 packets

RIP2B Unicast/Broadcast RIP Version 2 packets (Not Recommended)

RIP2M
Unicast/Multicast/Broadcast RIP packets (Migration)

NONE Disable sending RIP packets

RIP_RECEIVE_CONTROL:
A constant. Specifies that the keyword following is to be used as the RIP receive control for all interfaces. Possible receive controls are as follows:

RIP1 Receive RIP Version 1 packets

RIP2 Receive RIP Version 2 packets

ANY Receive any RIP Version 1 and 2 packets (Default)

NONE Disable receiving RIP packets

Figure 117 on page 777 is a sample OROUTED configuration file:

```
; EZARTPRF
;
; COPYRIGHT = NONE.
;
; Sample OROUTED profile.
; See the "IP Configuration Guide for more information"
;
; ---------------------------------------------------------------------
; RIP_SUPPLY_CONTROL specifies one of the following options on a
; router-wide basis:
;
; 1) RIP1  - Unicast/Broadcast RIP Version 1 packets (Default)
; 2) RIP2B - Unicast/Broadcast RIP Version 2 packets (Not Recommended)
; 3) RIP2M - Unicast/Multicast/Broadcast RIP packets (Migration)
; 4) RIP2  - Unicast/Multicast RIP Version 2 packets
; 5) NONE  - Disables sending RIP packets
;
; Note: If RIP2 is specified, the RIP Version 2 packets are multicast
;       over multicast-capable interfaces only.  No RIP packets are
;       sent over multicast-incapable interfaces.  For RIP2M, the RIP
;       Version 2 packets are multicast over multicast-capable
;       interfaces and RIP Version 1 packets are broadcast or unicast over
;       multicast-incapable interfaces.  For RIP2B, the RIP Version
;       2 packets are broadcast or unicast; this option is not recommended since
;       host route misinterpretations by adjacent routers running RIP
;       Version 1 can occur.  For this reason, RIP2B may become
;       obsolete in a future release.  For point-to-point interfaces
;       that are non-broadcast and multicast-incapable, the RIP
;       Version 2 packets are unicast.

RIP_SUPPLY_CONTROL: RIP1

; RIP_RECEIVE_CONTROL specifies one of the following options on a
; router-wide basis:
;
; 1) RIP1   - Receive RIP Version 1 packets only
; 2) RIP2   - Receive RIP Version 2 packets only
; 3) ANY    - Receive any RIP Version 1 and 2 packets (Default)
; 4) NONE   - Disables receiving RIP packets

RIP_RECEIVE_CONTROL: ANY

; RIP2_AUTHENTICATION_KEY specifies a plain text password containing
; up to 16 characters.  The authentication key must be enclosed in
; double quotes.  The key is used on a server-wide basis and can
; contain mixed case and blank characters.  The key will be used to
; authenticate RIP Version 2 packets and be included in the
; RIP responses for authentication by adjacent routers running RIP
; Version 2.  A null key (either no key is specified or "" is
; specified) indicates that authentication is disabled.  For
; maximum security, set RIP_SUPPLY_CONTROL and RIP_RECEIVE_CONTROL
; to RIP2.  This will discard RIP1 and unauthenticated RIP2 packets.

RIP2_AUTHENTICATION_KEY:
```

Figure 117. Sample OROUTED configuration file

Step 2: Update configuration statements in PROFILE.TCPIP

To ensure that UDP port 520 is reserved for OROUTED, also add the name of the member containing the OROUTED cataloged procedure to the PORT statement in PROFILE.TCPIP:

```
PORT
   520 UDP OROUTED
```

In addition, configure the BSDROUTINGPARMS statements with your routing information. The BEGINROUTES or GATEWAY statement is not used and should be removed or commented from PROFILE.TCPIP.

If a static route must be coded in the GATEWAY statement in the TCP/IP Profile, a corresponding external route must be coded in ORouteD's /etc/gateways. This will inform ORouteD that these routes are locally defined and are not to be changed; in addition, by the external definition for a place holder in ORouteD's routing table, ORouteD will not attempt to delete or add equivalent routes from the RIP updates.

For multipath routes with common destination addresses reachable over multiple interfaces and/or multiple gateways or nexthops, code a corresponding external route from one of the multipath routes in /etc/gateways. Because ORouteD supports only one route to a destination and because the nexthop or gateway address is irrelevant in external route definitions, ORouteD will use the external route definition to represent any static route sharing a common destination address and prevents static route overrides from competing RIP routes.

Code the following on the IPCONFIG statement in PROFILE.TCPIP:

```
IPCONFIG IGNOREREDIRECT DATAGRAMFWD
```

Do not specify the no forwarding (NOFWD) option. To enable variable subnetting, add the VARSUBNETTING option. If OROUTED is to be used to either send or receive RIP version 2 packets, the VARSUBNETTING option must be specified in the TCPIP profile. To enable outbound source VIPA, add the SOURCEVIPA option.

Refer to *z/OS Communications Server: IP Configuration Reference* for descriptions and examples of these statements.

Note: If you want to be able to start OROUTED from the z/OS shell, use the special name OMVS as follows:

```
PORT 520 UDP OMVS
```

This enables the entire "OMVS job group" (that is, all z/OS shell users). Only RACF-authorized users can start OROUTED.

Step 3: Update the resolver configuration file

The resolver configuration file or data set contains keywords that are used by OROUTED. Two important keywords in the resolver file are DATASETPREFIX and TCPIPjobname. The value assigned to DATASETPREFIX will determine the high-level qualifier (*hlq*). The *hlq* is then used in the search order for other configuration files. If no DATASETPREFIX keyword is found in the resolver configuration data set or file, a default of TCPIP is used. In a CINET environment, the value assigned to TCPIPjobname will be used as the name of the stack with which OROUTED attempts to establish a connection. In an INET environment, it is not necessary to set TCPIPjobname, but if it is set, it must be set to ″INET″.

For a description of the search order used by the resolver to locate the resolver configuration data set or file, see "Resolver configuration files" on page 27.

Step 4: Update the OROUTED cataloged procedure (optional)

If OROUTED is to be started by a procedure, update the cataloged procedure OROUTED by copying the sample in *hlq*.SEZAINST(OROUTED) to your system or recognized PROCLIB. Specify OROUTED parameters and change the data set names as required to suit your local configuration.

Note: When using PGM=BPXBATCH to start OROUTED, STDOUT and STDERR cannot be directed to any SYSOUT class; they must be directed to an HFS file.

OROUTED cataloged procedure

A data set or file may be used to set the environment variables for an invocation of OROUTED. This file is specified on the STDENV statement. An example of its contents is included in the OROUTED cataloged procedure sample. For more information about the cataloged procedure, refer to the OROUTED chapter in *z/OS Communications Server: IP Configuration Reference*. For more complete information about STDENV, refer to *z/OS Communications Server: IP User's Guide and Commands*.

Step 5: Specify the OROUTED port number in the SERVICES file

The services data set or file contains the relationship between service names (servers) and port numbers in the z/OS UNIX environment. The portion of the services file relevant to OROUTED is shown in Figure 118. The data set or file must exist for OROUTED to run. The following search order is used to find the services data set or file:

1. /etc/services
2. *userid*.ETC.SERVICES, where *userid* is the user ID that is associated with the current security environment (address space or task/thread).
3. *hlq*.ETC.SERVICES

```
# Start of IBM added services ...
route        520/udp       router routed
```

Figure 118. Sample portion of services file

Step 6: Configure the gateways file or data set (optional)

The OROUTED server queries the network and dynamically builds routing tables from routing information transmitted by other routers that are directly connected to the network. The gateways file or data set is used to further configure the routing tables.

Note: The gateways file or data set is not related to the GATEWAY statement used in the PROFILE.TCPIP data set.

The OROUTED server uses the following search order to locate the GATEWAYS configuration data set or file:

1. If the environment variable GATEWAYS_FILE has been defined, OROUTED uses this value as the name of an MVS data set (//'mvs.dataset.name') or HFS file (/dir/subdir/file.name) to access the gateways file
2. /etc/gateways
3. *hlq*.ETC.GATEWAYS

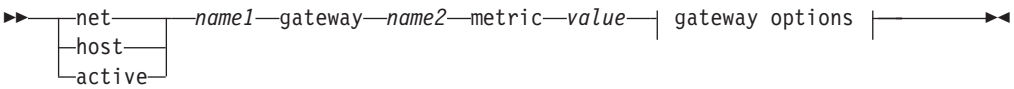
A sample gateways file is provided in *hlq*.SEZAINST(EZARTGW).

A `passive` entry in the gateways file or data set is used to add a route to a part of the network that does not support RIP. An `external` entry in the gateways file or data set indicates a route that should never be added to the routing tables. If another RIP server offers this route to your host, the route is discarded and not added to the routing tables. An `active` entry indicates a gateway that can only be reached through a network that does not allow or support link-level broadcasting or multicasting.

Syntax rules

- The maximum LRECL allowed for the ETC.GATEWAY data set is 999.
- Keywords can be specified in mixed case.
- Blanks and comments are supported in the gateways file or data set. Comments are identified by a semicolon in column 1.
- There should be no sequence numbers in the data set.

The syntax for the gateways file or data set is:



gateway options:

net
Indicates the route goes to a network.

host
Indicates the route goes to a specific host.

active
Indicates that the route to the gateway will be treated as a network interface. Active gateways are routers that are running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

name1
Can be either a symbolic name or the IP address of the destination network or host. If an IP address is specified, it must be in the standard dotted decimal notation. All numbers will be interpreted as decimal values only. No hexadecimal or octal notation will be accepted.

name1 must be specified as “active” if this is for an active gateway. The last entry in the data set must specify an active gateway.

gateway
A constant. The parameters that follow this keyword identify the gateway or router for this destination.

name2
Can be either a symbolic name or the IP address of the gateway or router for this destination. If an IP address is specified, it must be in the standard dotted

decimal notation. All numbers will be interpreted as decimal values only. No hexadecimal nor octal notation will be accepted.

metric
A constant. The value that follows this keyword is the hop count to the destination host or network.

value
The hop count to this destination. This number is an integer in the range of 0 through 16, where 16 (infinity) indicates the network cannot be reached.

passive
A passive gateway does not exchange routing information. Information about the passive gateway is maintained in the local routing tables indefinitely and is only local to this OROUTED server. Passive gateway entries for indirect routes are not included in any routing information that is transmitted. Directly connected passive routes are included.

external
An external gateway parameter indicates that entries for this destination should never be added to the routing table. The OROUTED server discards any routes for this destination that it receives from other routers. Only the destination field is significant. The gateway parameter is ignored, but you must specify a routing interface in the network. The metric field is ignored.

active
Active gateways are treated as network interfaces. Active gateways are routers that are running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

mask
A constant. The value that follows this keyword is the subnet mask for the route.

subnetmask
A bit mask (expressed in dotted-decimal form) defining the subnetwork mask for a network route. The bits must be contiguous in the network portion of the *subnetmask*. If the *subnetmask* is not specified, OROUTED will default the subnetwork mask to an interface subnetwork mask that matches the route's network. If there is no interface match, then the network class mask for the route is used.

Note: For more information on passive, external, and active gateways, see *z/OS Communications Server: IP Configuration Reference*.

The following example shows the contents of a gateways file or data set containing multiple entries:

```
net    acmenet       gateway  gateway.acme.com  metric 5  passive
host   vm3.ibm.com   gateway  9.67.43.126       metric 6  passive
host   bad.host      gateway  9.67.113.1        metric 1  external
active active        gateway  9.3.1.110         metric 3  active
net    0.0.0.0       gateway  9.67.112.1        metric 1  passive
```

In the first entry, the route indicates that `acmenet` can be reached through the gateway `gateway.acme.com`, and that it is 5 hops away.

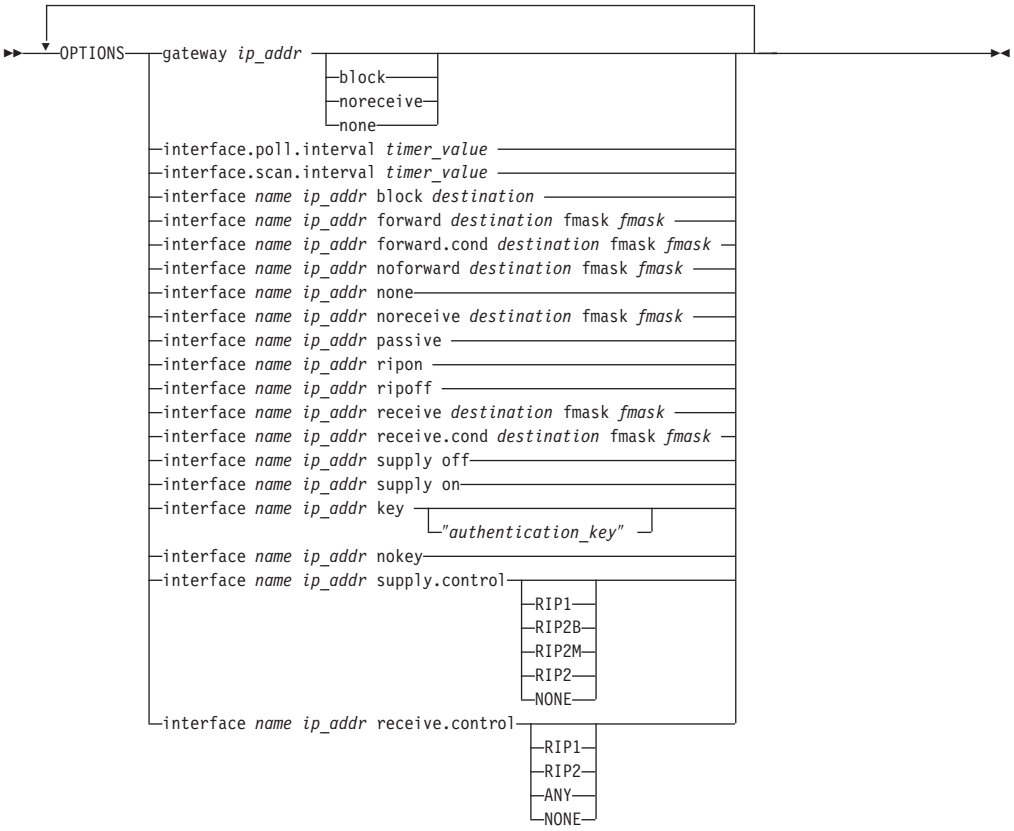
In the second entry, the route indicates that `vm3.ibm.com` can be reached through the gateway `9.67.43.126`, and that it is 6 hops away.

In the third entry, the external gateway parameter indicates that routes for the host `bad.host` should not be added to the routing tables, and that routes received from other OROUTED servers for `bad.host` should not be accepted.

The fourth entry shows an active gateway.

The fifth entry shows a default route to the destination gateway 9.67.112.1.

The syntax for the OPTIONS statement for the gateways file or data set is:



gateway
A constant. The value that follows this keyword identifies the gateway or router.

interface.scan.interval
Specifies the time interval in seconds for the interface scan interval. OROUTED uses this timer value to rescan existing interfaces for up/down status, new interfaces, and new HOME lists. New interfaces and HOME lists are dynamically added using VARY TCPIP,,CMD=OBEYFILE commands.
timer_value
The range is from 30 to 180 seconds in multiples of 30 seconds. The default is 60 seconds.

interface.poll.interval
Specifies the time interval in seconds for the interface poll interval. OROUTED uses this timer value to check existing interfaces for up/down status only. Triggered updates are issued during interface outages to inform adjacent routers of unreachable routes so that alternative routes can be discovered.

timer_value
The range is from 15 to 180 seconds in multiples of 15 seconds. The default is 30 seconds.

interface
A constant

name
Specifies the name of the interface as used in the HOME list. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all interface names.

ip_addr
Specifies the internet address of the interface associated with the interface name. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all internet addresses of the interfaces.

block
For the interface option, specifies that the *destination* route in the received broadcasts for this interface is to be ignored. For the gateway option, specifies that routing table RIP responses from this gateway are to be ignored. This option is provided as a RIP input filter.

destination
Specifies the destination route in network, subnetwork, or host format. A specification of an asterisk (*) indicates that all destination routes to be used with the noforward and noreceive options. This serves as a ″blackhole″ filter option which can be used to filter out all routes from RIP packets to be sent or received over an interface and allow routes with specified forward and receive filters to be used.

fmask
Specifies the optional route filter mask.

forward
Specifies that the *destination* route in the RIP responses is to be forwarded to this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

forward.cond
Specifies that the *destination* route is to be forwarded to this interface only when the interface is active. In case of an interface outage, OROUTED will include the *destination* route in the RIP responses to other active interfaces. After recovery of an interface outage, ORouteD will resume to sending the destination route over this interface only. This option is provided as a RIP output filter and can be used for inbound and outbound traffic splitting.

noforward
Specifies that the destination route in the RIP responses is not to be forwarded. This option is provided as a RIP output filter.

noreceive
See description for block.

passive
Same as ripoff.

receive
Specifies that the *destination* route is to be received over this interface only. If it is received over any other interface, the route is discarded. This option is provided as a RIP input filter.

receive.cond
Specifies that the *destination* route is to be received over this interface only when the interface is active. In case of an interface outage, OROUTED will allow the *destination* route in the RIP responses to be received over other active interfaces. This option is provided as a RIP input filter and can be used for inbound and outbound traffic splitting.

ripoff
Specifies that RIP is disabled for this interface. OROUTED will not send or receive RIP updates. This option is provided as a RIP input and output filter.

ripon
Specifies that RIP is enabled for the interface. This is the default for all interfaces. This option should be used when RIP has been previously disabled for an interface with the ripoff option, but is now required to be enabled for that interface.

supply off
Specifies that supplying RIP responses is disabled for this interface. OROUTED will not send, but continues to receive RIP responses. This option is provided as a RIP output filter.

supply on
Specifies that supplying RIP responses is enabled for this interface. This option is provided as a RIP output filter.

none
For the interface option, specifies that any RIP filter options for this interface are to be turned off or reset. If an asterisk (*) is specified for the interface *name* and *ip_addr*, all options will be cleared from all interfaces. For the gateway option, specifies that any RIP filter options for this gateway are to be turned off or reset. If an asterisk (*) is specified for the internet address, all gateway entries with gateway options will be cleared.

key
Specifies a plain text password containing up to 16 characters for the authentication key to be used for this interface and is used to override the router-wide setting defined in the OROUTED profile data set. The key must be enclosed in double quotes for the delimiters and can contain mixed case and blank characters. A no key or null key (″″) specification indicates that the router-wide key will be used as the default.

authentication_key
An authentication key containing up to 16 characters to be used for this interface and is used to override the OROUTED profile setting. The key must be enclosed in double quotes. The key will start with the first character past the first quotation mark and end at the last character before the last quotation mark on the line.

nokey
Specifies that authentication is disabled for this interface even though the router-wide specification from the OROUTED profile is defined.

supply.control
A constant. Specifies that the keyword following is to be used as the RIP supply control for this interface and is used to override the OROUTED profile setting. Possible supply controls are as follows:

RIP1 Unicast/Broadcast RIP Version 1 packets (Default)

RIP2B Unicast/Broadcast RIP Version 2 packets (Not Recommended)

RIP2M
Unicast/Multicast/Broadcast RIP packets (Migration)

RIP2 Unicast/Multicast RIP Version 2 packets

NONE Disable sending RIP packets

receive.control
A constant. Specifies that the variable following is to be used as the RIP receive control for this interface and is used to override the OROUTED profile setting. Possible receive controls are as follows:

RIP1 Receive RIP Version 1 packets

RIP2 Receive RIP Version 2 packets

ANY Receive any RIP Version 1 and 2 packets (Default)

NONE Disable receiving RIP packets

The following example shows the options entries of a gateways file or data set:

```
options interface.scan.interval 90
options interface.poll.interval 15
options interface ETH1 10.1.1.1 passive
options interface ETH1 10.1.1.1 supply off
options interface TR1 9.67.112.25 forward 11.0.0.0
options interface TR1 9.67.112.25 forward.cond 12.0.0.0
options interface TR1 9.67.112.25 block 9.1.0.0
options interface TR1 9.67.112.25 supply.control rip1
options interface ETH1 10.1.1.1 receive.control rip2
options interface ETH1 10.1.1.1 key
options interface CTC0  9.67.114.22 key "shredder"
options interface ETH1 10.1.1.1 none
options interface * * none
options gateway 9.2.1.4 noreceive
options gateway 9.2.1.4 none
options gateway * none
```

Step 7: Configure and start syslogd

If not already started, configure and start syslogd. For more information on syslogd, see “Logging of system messages” on page 39. Otherwise, all output messages will go to the operator system console.

Step 8: RACF-authorize user IDs

To control which users can start OROUTED (and thus reduce risk of an unauthorized user starting it and affecting the contents of the routing table), the appropriate users must be RACF-authorized to the entity MVS.ROUTEMGR.OROUTED. To do this, the following commands must be entered from a RACF user ID, substituting the authorized user ID on the ID (userid) parameter:

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OROUTED) UACC(NONE)
PERMIT MVS.ROUTEMGR.OROUTED ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

OROUTED parameters

OROUTED accepts the command line parameters listed below. These parameters are valid when starting the program from either an MVS procedure or from the z/OS shell. They are also valid when modifying OROUTED with the MODIFY command. For information on using the MODIFY command, see *z/OS Communications Server: IP System Administrator's Commands*.

-c <filename.filetype>
OROUTED configuration files are processed and a file which can be used as an OMPROUTE configuration file is created. If start parameters include -c, OROUTED will terminate once the conversion file has been created. If modify parameters include -c, OROUTED will remain active . Filename.filetype is an option for this parameter. If -c is specified with a file name as a parameter for modify of OROUTED, the resulting file name is automatically upper cased regardless of the original input. The default output file is CNVROUTED.PROFILE in the /tmp directory. The user must put this file in the appropriate directory or data set for use by OMPROUTE. The converted profile should be analyzed to determine if comments direct changes to the TCP/IP profile. If suggested by comments, customer should take appropriate action to update TCP/IP profile. If the file name already exists, OROUTED will not create a conversion file and the user must specify -c again with a new output file name or erase the existing file. When starting OROUTED, this parameter is allowed with all other start parameters.

-del
All dynamic routes are deleted from the routing table upon initialization of OROUTED. By default, OROUTED deletes only RIP routes at startup.

-d Enables printing internal debug information to standard output. This option should only be used to debug problems. When this option is specified, the -ep parameter is set internally.

-dp
Traces packets to and from adjacent routers and received and sent RIP packets. Packets are displayed in data format. Output is written to standard output.

-ep
Enable display of program print statements to standard output and standard error. Information can be saved to a file by redirecting standard output to a file using the '>' operator. If this option is specified, and the program is started in the background from a z/OS shell, the userid will not be able to exit the shell until the program has ended. This parameter is required is you start OROUTED with a procedure and PGM=OROUTED. It is not required if you start OROUTED with a proc and PGM=BPXBATCH.

-g Enables the default router. When this option is specified, OROUTED will add a default route to its routing information and propagate it over all local interfaces. If the adjacent routers add the default route to their routing tables, OROUTED will receive all unknown packets from them and funnel them to a destination router, provided that a default route is defined. If you use this option, we recommend that you define a default route to a destination router in the gateways file or data set. See "Configuring a default route" on page 792.

Note: Do not use this option if default routes are to be learned dynamically from adjacent routers.

-h Include host routes in addition to network routes for the RIP responses.

Adjacent routers must be able to receive host routes to prevent NETWORK UNREACHABLE problems from occurring.

-hv
Include only VIPA host routes in addition to network routes for the RIP responses. Adjacent routers must be able to receive host routes' otherwise, network or subnetwork portions of VIPA addresses must be unique for each z/OS TCP/IP stack.

-q Suppresses supplying routing information.

-sd
Supply default route only. When this option is specified, the -g parameter is set internally. This option is provided as a RIP output filter.

-sdv (or -svd)
Supply network-specific VIPA routes and default routes only. See parameter descriptions for -sv and -sd. This option is provided as a RIP output filter.

-sl Supply local (directly-connected) routes only. This option is provided as a RIP output filter.

-st Supply triggered updates only. Similar to the -q parameter except that OROUTED will supply network unreachable routing information during interface outages so that adjacent routers can recover by switching to different routes rather than relying on three-minute timeouts. This option is provided as a RIP output filter.

-sv
Supply network-specific VIPA routes only. Recommended usage is when multiple network adapters in a z/OS TCP/IP stack are in the same network; otherwise, network connectivity problems will occur. This option is provided as a RIP output filter.

-svd
Similar to -sdv parameter.

-svh (or -shv)
Supply VIPA (network-specific and host) routes only. This option is provided as a RIP output filter.

-t Activates tracing of actions by the OROUTED server.

-t -t
Activates tracing of actions and packets sent or received.

-t -t -t
Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced and are displayed whenever an interface becomes inactive.

-t -t -t -t
Activates tracing of actions, packets sent or received, packet history, and packet contents. The packet displays the RIP network routing information.

Table 28 on page 788 shows how the above parameters affect the advertising algorithm for routes in RIP responses to adjacent routers. The parameters can be used as router-wide RIP output filters. To configure interface-wide RIP input and output filters, see the OPTIONS statement in the GATEWAYS configuration data set or file.

Table 28. ORouteD parameters

Parameter	VIPA host routes	Host routes (direct and indirect)	VIPA network routes	Direct (local) network routes	Indirect network routes	Default routes	Unreachable routes
-g			Yes	Yes	Yes	Yes	Yes
-h	Yes	Yes	Yes	Yes	Yes		Yes
-hv	Yes		Yes	Yes	Yes		Yes
-s			Yes	Yes	Yes		Yes
-sd						Yes	Yes
-sl			Yes	Yes			Yes
-sq or -q							
-st							Yes
-sv			Yes				Yes
-svd			Yes			Yes	Yes
-svh	Yes		Yes				Yes
None			Yes	Yes	Yes		Yes

Specifying parameters

If OROUTED is to be started from an MVS procedure, add your parameters to PARM= in the OROUTED cataloged procedure. For example:

```
//OROUTED  PROC PARMS='-ep -t -t'
//OROUTED  EXEC PGM=OROUTED,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON)',
//       'ENVAR("_CEE_ENVFILE=DD:STDENV")',
//       '/&PARMS')
```

If OROUTED is to be started from a z/OS shell command line, enter the parameters on the z/OS shell command line.

For either method of starting OROUTED, the following apply:

- Each parameter is separated by a blank.
- Parameters can be specified in mixed case.

Starting OROUTED

In a CINET environment, OROUTED will attempt to connect to a stack name whose name is determined by the *TCPIPJobname* keyword from the resolver configuration data set or file. The *TCPIPJobname* must match the NAME field for ENTRYPOINT(EZBPFINI) in the BPXPRMxx member you used to start OMVS. For information on configuring multiple TCP/IP instances as z/OS UNIX CINET physical file systems, see “Considerations for multiple instances of TCP/IP” on page 54.

In configurations with multiple stacks, a copy of OROUTED must be started for each stack that requires OROUTED services. To associate OROUTED with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique *TCPIPjobname*. A unique gateways file can be associated with each copy of OROUTED by defining the environment variable

GATEWAYS_FILE. In the example in Figure 119 for the z/OS shell, there are two active stacks with the names TCP_A and TCP_B. First the environment variable RESOLVER_CONFIG is set to point to the file that defines the *TCPIPjobname* TCP_A, then the environment variable GATEWAYS_FILE is set to point to the gateways file /etc/gateways.tcp_a, and then OROUTED is started for that stack. Next, RESOLVER_CONFIG is set to point to another configuration file that defines *TCPIPjobname* TCP_B, then the environment variable GATEWAYS_FILE is set to point to the gateways file /etc/gateways.tcp_b, and then OROUTED is started for that stack.

```
#  export RESOLVER_CONFIG=/u/user105/tcp_a.conf          !point to TCP_A resolver file
#  export GATEWAYS_FILE=/etc/gateways.tcp_a              !point to TCP_A gateways file
#  export _BPX_JOBNAME=RDA                               !set procname to RDA
#  export ROUTED_PROFILE=/u/user105/rda.profile          !set routed profile
#  orouted&                                              !start orouted in the background
#  export RESOLVER_CONFIG=/u/user105/tcp_b.conf          !point to TCP_B resolver file
#  export GATEWAYS_FILE=/etc/gateways.tcp_b              !point to TCP_77 bB gateways file
#  export _BPX_JOBNAME=RDB                               !set procname to RDB
#  export ROUTED_PROFILE=/u/user105/rda.profile          !set routed profile
#  orouted&                                              !start orouted in the background
```

Figure 119. Example commands to start multiple copies of OROUTED

When running from an MVS procedure, the environment variables can be set by using the STDENV DD statement in the procedure used to start OROUTED. For an example of using the STDENV DD statement, see “OROUTED cataloged procedure” on page 779.

Configuration examples

This section contains examples for configuring the OROUTED server. The following example illustrates a OROUTED configuration.

Configuring a passive route

In Figure 120, assume that your z/OS server is `host1` and is running an OROUTED server. The other two hosts, `host2` and `host3`, are not running a RIP server. Your OROUTED server does not learn a route to `host3`, because `host2` is not running a RIP server. Your OROUTED server sends routing updates to `host3` over the link to `host2` but never receives a routing update from `host2`. After 180 seconds, your OROUTED server deletes the route to `host2`. This problem is inherent to the RIP protocol and cannot be prevented.

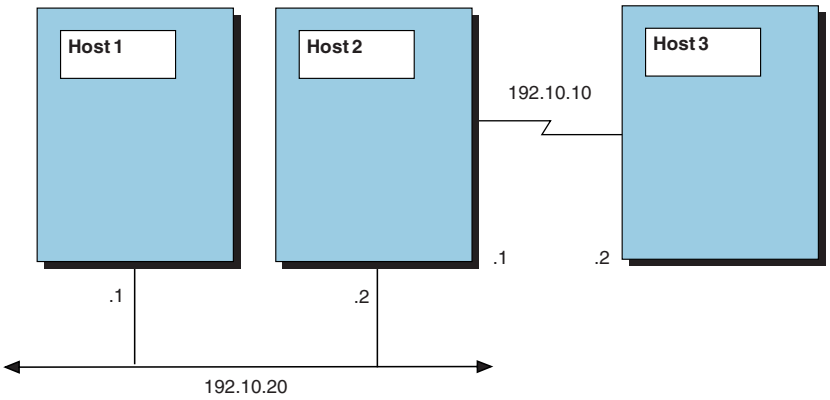


Figure 120. OROUTED configuration example

To solve the problem, you should add a `passive` route to this host in the gateways file or data set. You can use either of the following gateway statements:

```
host  host3        gateway  host2        metric 2  passive

host  192.10.10.2  gateway  192.10.20.2  metric 2  passive
```

Similarly, if `host2` is not running a RIP server, you can define a directly-connected `passive` route as follows:

```
host  host2        gateway  host1        metric 1  passive
```

A directly-connected passive route is one where the gateway address or name is one of the local interfaces in the HOME list.

Assume that your z/OS server is now `host2` and is running a OROUTED server. `host1` is also running a RIP server, but `host3` is not. Your OROUTED server sends routing information updates to `host3` over the link to `host3` but never receives a routing update from `host3`. After 180 seconds, your OROUTED server deletes the route to `host3`.

You should add a passive route to this host as follows:

```
host  host3        gateway  host2        metric 1  passive
```

`host1` cannot reach `host3` unless a passive routing entry is added to `host1`. For example:

```
host   host3    gateway  host2  metric  2  passive
```

or

```
host  192.10.10.2  gateway  192.10.20.2  metric  2  passive
```

Configuring an external route

In Figure 120, assume that your z/OS server is again `host1`, which is running an OROUTED server. The other two hosts, `host2` and `host3`, are also running RIP servers. Your OROUTED server normally learns a route to `host3` from `host2`, because `host2` is running a RIP server. You might not want `host1` to route to `host3` for security reasons. For example, a university might want to prevent student hosts from routing to administrative hosts.

To prevent your OROUTED server from adding a route to host3, add an external route to the gateways file or data set. You can use either of the following gateway statements:

```
host  host3        gateway  host2        metric 2  external

host  192.10.10.2  gateway  192.10.20.2  metric 2  external
```

Configuring an active gateway

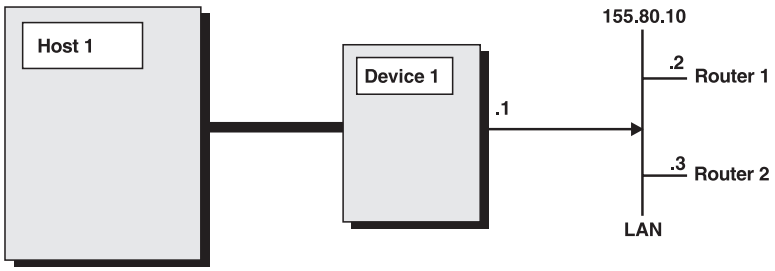


Figure 121. Configuring an active gateway

As shown in Figure 121, assume that your MVS host is host1, which is running an OROUTED server and that device1 is a network attachment device that does not support link-level broadcasting or multicasting or one that does not support ARP processing (for example, HYPERchannel and ATM). Also, assume that host1 is channel-connected to device1 as a hyperchannel device and that there are routers router1 and router2 on the local area network. Because the IP addresses for router1 and router2 are unknown by host1, they have to be manually configured in host1 for OROUTED to communicate with them. Configuring active gateways for router1 and router2 as remote network interfaces enables OROUTED to send RIP responses to the target addresses.

Include the following definitions in *hlq*.PROFILE.TCPIP for host1, router1, and router2:

1. Specify DEVICE and LINK statements for the hyperchannel. For example:

   ```
   DEVICE HYPER1   HCH    CE2
   LINK   HYPER1A  HCH    2     HYPER1A
   ```

2. Add the TRANSLATE statement for the remote routers on the local area network attached to the hyperchannel device:

   ```
   TRANSLATE 155.80.10.2 HCH    HYPER1A
   TRANSLATE 155.80.10.3 HCH    HYPER1A
   ```

3. Add the hyperchannel link to the HOME statement for the assignment of local IP address:

   ```
   HOME
        155.80.10.1  HYPER1A
   ```

4. Add the hyperchannel link to the BSDROUTINGPARMS statement:

   ```
   BSDROUTINGPARMS false
         HYPER1A       16384      0      255.255.240.0    0
   ENDBSDROUTINGPARMS
   ```

Define active gateways for the remote routers in the OROUTED gateways file or data set:

```
active active gateway 155.80.10.2 metric 1 active
active active gateway 155.80.10.3 metric 1 active
```

From these active gateway addresses, OROUTED will use them as the destination addresses to send RIP responses to the remote routers. In addition, OROUTED will continue to receive RIP responses from the active gateways over the hyperchannel device.

Configuring a point-to-point link

The OROUTED server can manage point-to-point links as long as there are RIP services at both ends of the links. If a host router at the other end of a link is not running a RIP service, then passive routing must be configured in the OROUTED gateways data set for the link. See “Configuring a passive route” on page 789.

Configuring a default route

A default route is typically used on a gateway or router to an internet, or on a gateway or router that uses another routing protocol, whose routes are not reported to other local gateways or routers.

To configure a route to a default destination, add a default route using the passive route definition in the gateways file or data set. For example, if the default destination router has a gateway address 9.67.112.1, then add the following entry to the data set:

```
net  0.0.0.0  gateway  9.67.112.1  metric  1  passive
```

Only one default route to a destination gateway or router can be specified. OROUTED currently does not support multiple default routes.

Configuring ORouteD with Enterprise Extender

If running ORouteD with the Enterprise Extender on a z/OS TCP/IP stack connected to SNA via IUTSAMEH device, special configuration needs to be considered depending on the system environment. Assume that TOVTAM interface is the link name for the IUTSAMEH device and its home IP address is 9.2.1.1.

1. If there are no other z/OS TCP/IP stacks in this MVS image, disable RIP on the TOVTAM interface to prevent ORouteD from sending any routing information over this interface. For example, in the ORouteD gateways file or data set:

   ```
   options interface TOVTAM 9.2.1.1 ripoff
   ```

2. If one or more z/OS TCP/IP stacks exist in the MVS image, do the following:
   - Define the link characteristics for the TOVTAM interface in the BSDROUTINGPARMS statement of TCP/IP configuration profile. In this definition, specify a subnet mask and a zero IP destination address.

     For example, assume that the subnet mask for the TOVTAM interface is 255.255.255.0:

     ```
     ; link  maxmtu       metric   subnet mask     dest_addr
     TOVTAM  DEFAULTSIZE    0     255.255.255.0       0
     ```

     If using RIP1 or RIP2B on this interface and depending on its subnet mask, ORouteD will use a subnet-directed broadcast address to send the routing information to other z/OS TCP/IP stacks in this MVS image. If using RIP2 or RIP2M, ORouteD will use the RIP2 multicast address.
3. Repeat the above step for other z/OS TCP/IP stacks running with ORouteD and configured with IUTSAMEH devices in this MVS image. Ensure that the RIP

settings for the TOVTAM interfaces are identical so that the routing information exchanges will occur between z/OS TCP/IP stacks running ORouteD in this MVS image.

Configuring OROUTED with VIPA

For more information on configuration options with VIPA, see the following topics:

- Chapter 5, “Virtual IP Addressing” on page 209
- “Configuring static VIPAs for a z/OS TCP/IP stack” on page 213
- “Planning for static VIPA Takeover and Takeback” on page 215
- “Configuring OROUTED to split traffic with VIPA”

If Host Route advertising is not supported by adjacent routers (that is, inability to learn host routes), the following restrictions for VIPA addresses must be applied to benefit from fault tolerance support:

- If you use subnetting and VIPA addresses are in the same network as the physical IP addresses, the subnetwork portion of any VIPA addresses must not be the subnetwork portion of any physical IP addresses in the network. In this case, assign a new subnetwork for the VIPA address.
- If subnetting is not used on any physical interface, the network portion of any VIPA addresses must not be the network portion of any physical IP addresses in the network. In this case, assign a new network for the VIPA address, preferably a class C network address.

If Host Route advertising is supported by adjacent routers, the network or subnetwork portions of VIPA addresses can be the same across multiple z/OS TCP/IP stacks in the network. To enable Host Route advertising in OROUTED, specify option -h, -hv, or -svh.

Configuring OROUTED to split traffic with VIPA

The purpose of splitting traffic is to reduce traffic load on network attachments by controlling the inbound and outbound traffic. The following techniques can be used to produce traffic splitting effects with fault tolerance benefit:

- Using Interface Metric and VIPA To Split Inbound/Outbound Traffic

  In the multiple network attachments to the same network configuration, split inbound/outbound traffic can be achieved by configuring the metric on the primary interface to one higher than the secondary interface(s). From routing updates, an adjacent router uses the gateway of a secondary interface to reach the destination VIPA on the z/OS server because the route to the gateway has a shorter metric. The primary interface is used for outbound traffic and a secondary interface is used for inbound traffic. The traffic splitting will function as long as the primary and at least one secondary interfaces are active. For information on configuring an interface metric, see the *z/OS Communications Server: IP Configuration Reference*. A VARY TCPIP,,CMD=OBEYFILE command for the BSDROUTINGPARMS statement can be used to update an interface metric for a link. For an example of configuring a virtual device, see “Configuring OROUTED with VIPA”.

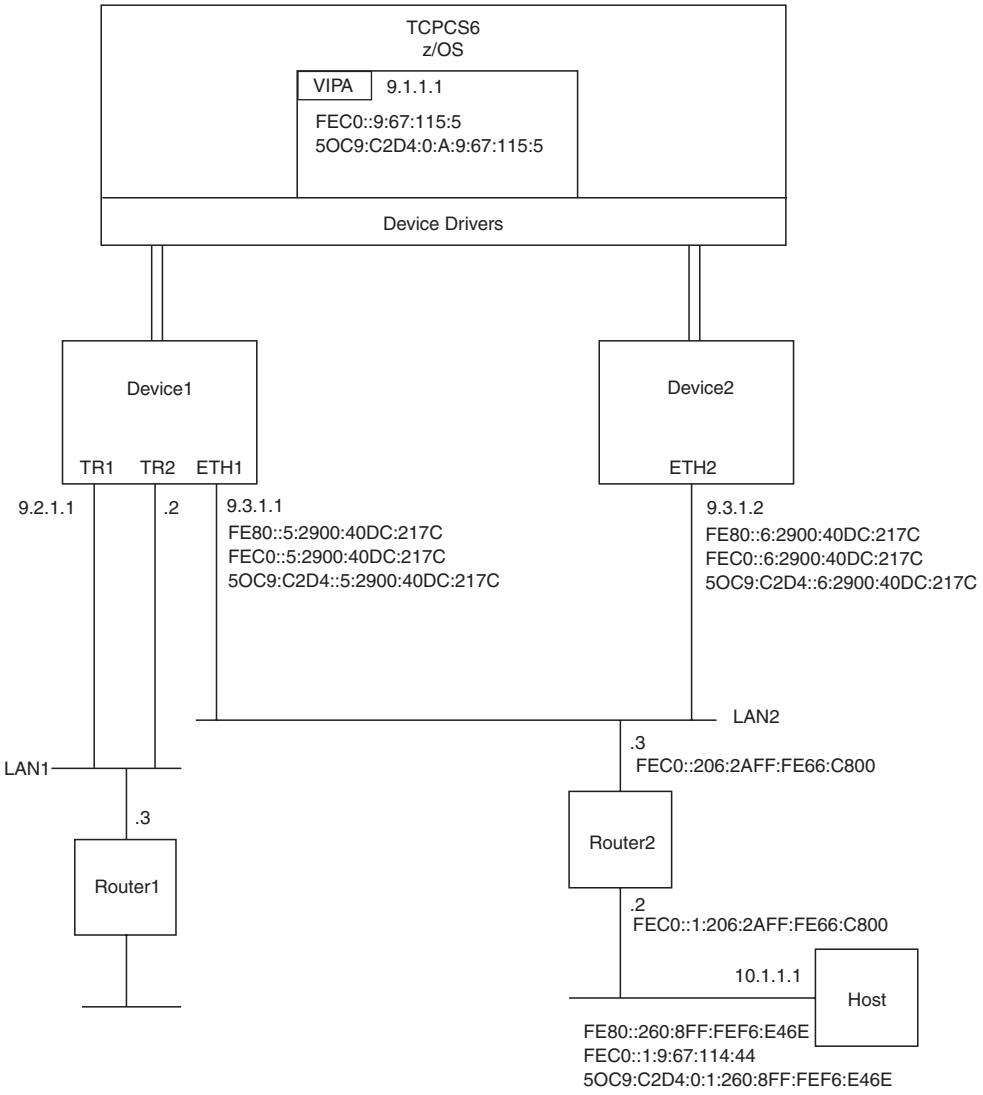


Figure 122. Single VIPA configuration. Sample configuration showing primary/multiple network attachments to the same LAN, VIPAs, and inbound/outbound traffic splitting.

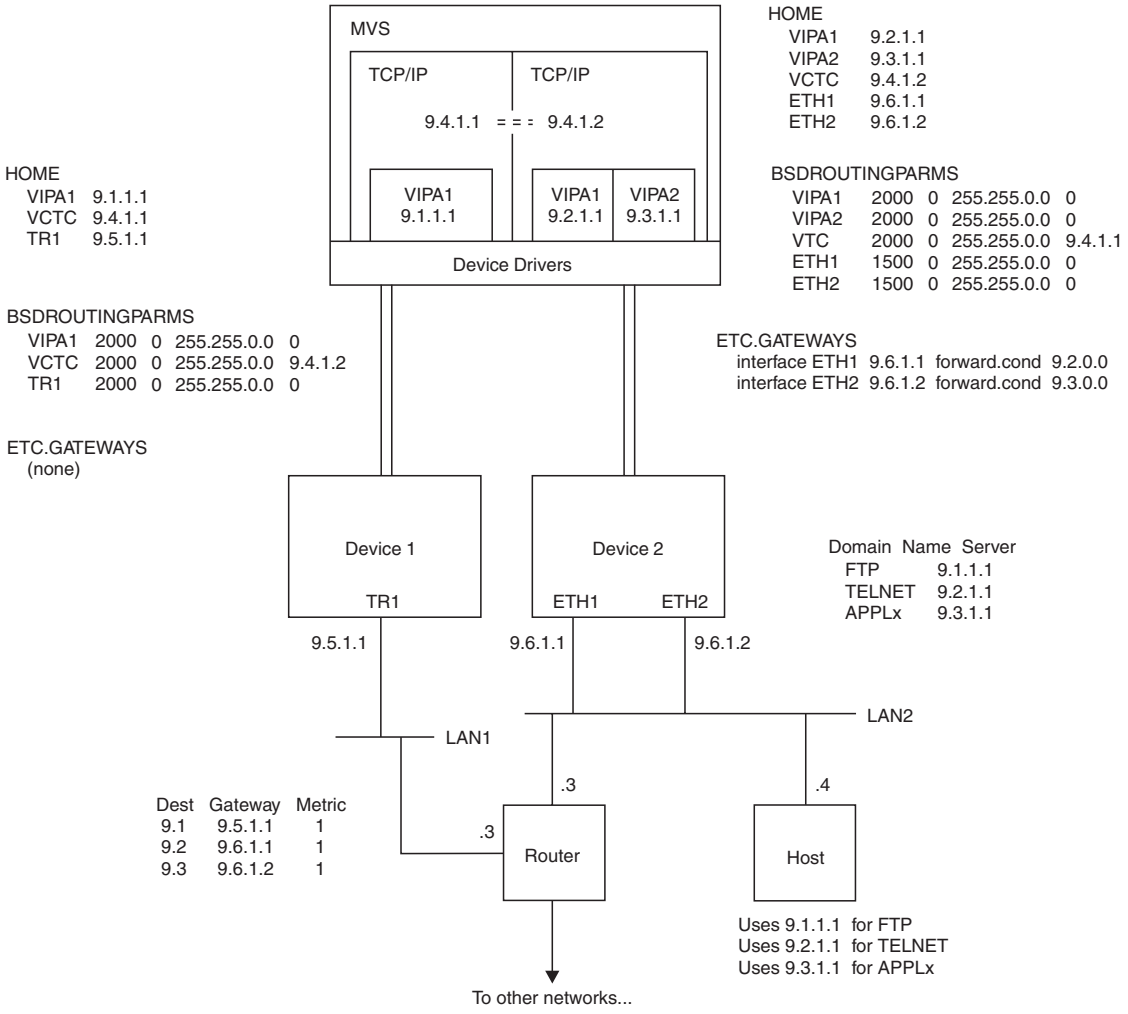


Figure 123. Multiple VIPA configuration. Sample configuration showing primary/multiple network attachments to the same LAN, VIPAs, and inbound/outbound traffic splitting.

- Using Route Forwarding and VIPA to Split Session Traffic

  With multiple VIPAs in one TCP/IP stack, a VIPA can be assigned to a particular interface so that the VIPA can be reserved for session traffic (for example, FTP or TELNET). This is accomplished by using the route forwarding option in OROUTED. From routing updates, an adjacent router will have multiple gateways to reach the VIPAs on the z/OS server. The adjacent router will use one gateway to reach one VIPA reserved for one type of session traffic and the other gateway to reach another VIPA reserved for another type of session traffic on the z/OS server. For fault tolerance, it is recommended that the conditional option of route forwarding be used. For information on route forwarding, see the options statement in “Step 6: Configure the gateways file or data set (optional)” on page 779. For an example of configuring a virtual device, see “Configuring OROUTED with VIPA” on page 793.

Configuring OROUTED with OSA-Express in QDIO mode

OSA-Express operating in QDIO mode (for example, gigabit Ethernet) supports multicasting but not broadcasting. Thus, RIP version 2, which relies on multicast, is recommended over RIP version 1, which relies on broadcast. No extra configuration is required for using RIP version 2.

If you must use RIP version 1, configure an active gateway entry in the OROUTED gateways file for each adjacent RIP-1 router in the network reachable through the QDIO interface.

Configuring OROUTED with HiperSockets

HiperSockets (iQDIO) supports multicasting but not broadcasting. Thus, RIP version 2, which relies on multicast, is recommended over RIP version 1, which relies on broadcast. No extra configuration is required for using RIP version 2.

OROUTED support for RIP version 1 for HiperSockets depends on how a HiperSocket was created.

If the HiperSocket was defined by DEVICE and LINK statements and was not created from IPCONFIG DYNAMICXCF, then you can use OROUTED for RIP version 1. Configure an active gateway entry in the OROUTED gateways file for each adjacent RIP-1 router in the iQDIO network.

If the HiperSocket was created from IPCONFIC DYNAMICXCF, then you cannot use OROUTED for RIP version 1 (an active gateway entry is not supported over multiple network interfaces that share the same IP address). However, OMPROUTE supports RIP version 1 over these HiperSockets. In the RIP_Interface statement in the OMPROUTE configuration file, configure the Neighbor parameter for each adjacent RIP-1 router in the iQDIO network.

Appendix F. Related protocol specifications (RFCs)

This appendix lists the related protocol specifications for TCP/IP. The Internet Protocol suite is still evolving through requests for comments (RFC). New protocols are being designed and implemented by researchers and are brought to the attention of the Internet community in the form of RFCs. Some of these protocols are so useful that they become recommended protocols. That is, all future implementations for TCP/IP are recommended to implement these particular functions or protocols. These become the *de facto* standards, on which the TCP/IP protocol suite is built.

These documents can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

where:

nnnn Is the RFC number.
TXT Is the text format.
PS Is the PostScript format.

You can see Internet drafts at http://www.ietf.org/ID.html. See "Draft RFCs" on page 804 for draft RFCs implemented in z/OS V1R4 Communications Server.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC nnnn` for text versions or a subject line of `RFC nnnn.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of `RFC INDEX`.

For more information, contact `nic@nic.ddn.mil`.

Many RFCs are available online. Hard copies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available using FTP from the NIC at the following Web address: http://www.rfc-editor.org/rfc.html.

Use FTP to download the files, using the following format:

```
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS
```

Many features of TCP/IP Services are based on the following RFCs:

RFC	Title and Author
768	*User Datagram Protocol* J.B. Postel
791	*Internet Protocol* J.B. Postel
792	*Internet Control Message Protocol* J.B. Postel
793	*Transmission Control Protocol* J.B. Postel
821	*Simple Mail Transfer Protocol* J.B. Postel

822 *Standard for the Format of ARPA Internet Text Messages* D. Crocker

823 *DARPA Internet Gateway* R.M. Hinden, A. Sheltzer

826 *Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.Bit Ethernet Address for Transmission on Ethernet Hardware* D.C. Plummer

854 *Telnet Protocol Specification* J.B. Postel, J.K. Reynolds

855 *Telnet Option Specification* J.B. Postel, J.K. Reynolds

856 *Telnet Binary Transmission* J.B. Postel, J.K. Reynolds

857 *Telnet Echo Option* J.B. Postel, J.K. Reynolds

858 *Telnet Suppress Go Ahead Option* J.B. Postel, J.K. Reynolds

859 *Telnet Status Option* J.B. Postel, J.K. Reynolds

860 *Telnet Timing Mark Option* J.B. Postel, J.K. Reynolds

861 *Telnet Extended Options—List Option* J.B. Postel, J.K. Reynolds

862 *Echo Protocol* J.B. Postel

863 *Discard Protocol* J.B. Postel

864 *Character Generator Protocol* J.B. Postel

877 *Standard for the Transmission of IP Datagrams over Public Data Networks* J.T. Korb

885 *Telnet End of Record Option* J.B. Postel

896 *Congestion Control in IP/TCP Internetworks* J. Nagle

903 *Reverse Address Resolution Protocol* R. Finlayson, T. Mann, J.C. Mogul, M. Theimer

904 *Exterior Gateway Protocol Formal Specification* D.L. Mills

919 *Broadcasting Internet Datagrams* J.C. Mogul

922 *Broadcasting Internet Datagrams in the Presence of Subnets* J.C. Mogul

950 *Internet Standard Subnetting Procedure* J.C. Mogul, J.B. Postel

952 *DoD Internet Host Table Specification* K. Harrenstien, M.K. Stahl, E.J. Feinler

959 *File Transfer Protocol* J.B. Postel, J.K. Reynolds

974 *Mail Routing and the Domain Name System* C. Partridge

1006 *ISO Transport Service on top of the TCP Version 3* M.T.Rose, D.E. Cass

1009 *Requirements for Internet Gateways* R.T. Braden, J.B. Postel

1011 *Official Internet Protocols* J. Reynolds, J. Postel

1013 *X Window System Protocol, Version 11: Alpha Update* R.W. Scheifler

1014 *XDR: External Data Representation Standard* Sun Microsystems Incorporated

1027 *Using ARP to Implement Transparent Subnet Gateways* S. Carl-Mitchell, J.S. Quarterman

1032 *Domain Administrators Guide* M.K. Stahl

1033 *Domain Administrators Operations Guide* M. Lottor

1034 *Domain Names—Concepts and Facilities* P.V. Mockapetris

1035 *Domain Names—Implementation and Specification* P.V. Mockapetris

1042 *Standard for the Transmission of IP Datagrams over IEEE 802 Networks* J.B. Postel, J.K. Reynolds

1044 *Internet Protocol on Network System's HYPERchannel: Protocol Specification* K. Hardwick, J. Lekashman

1055 *Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP* J.L. Romkey

1057 *RPC: Remote Procedure Call Protocol Version 2 Specification* Sun Microsystems Incorporated

1058 *Routing Information Protocol* C.L. Hedrick

1060 *Assigned Numbers* J. Reynolds, J. Postel

1073 *Telnet Window Size Option* D. Waitzman

1079 *Telnet Terminal Speed Option* C.L. Hedrick

1091 *Telnet Terminal-Type Option* J. VanBokkelen

1094 *NFS: Network File System Protocol Specification* Sun Microsystems Incorporated

1096 *Telnet X Display Location Option* G. Marcy

1101 *DNS encoding of network names and other types* P.V. Mockapetris

1112 *Host Extensions for IP Multicasting* S. Deering

1118 *Hitchhikers Guide to the Internet* E. Krol

1122 *Requirements for Internet Hosts—Communication Layers* R.T. Braden

1123 *Requirements for Internet Hosts—Application and Support* R.T. Braden

1155 *Structure and Identification of Management Information for TCP/IP-Based Internets* M.T. Rose, K. McCloghrie

1156 *Management Information Base for Network Management of TCP/IP-Based Internets* K. McCloghrie, M.T. Rose

1157 *Simple Network Management Protocol (SNMP)* J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin

1158 *Management Information Base for Network Management of TCP/IP-based internets: MIB-II* M.T. Rose

1179 *Line Printer Daemon Protocol* The Wollongong Group, L. McLaughlin III

1180 *TCP/IP Tutorial* T.J. Socolofsky, C.J. Kale

1183 *New DNS RR Definitions* C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris, (Updates RFC 1034, RFC 1035)

1184 *Telnet Linemode Option* D. Borman

1187 *Bulk Table Retrieval with the SNMP* M.T. Rose, K. McCloghrie, J.R. Davin

1188 *Proposed Standard for the Transmission of IP Datagrams over FDDI Networks* D. Katz

1191 *Path MTU Discovery* J. Mogul, S. Deering

1198 *FYI on the X Window System* R.W. Scheifler

1207 *FYI on Questions and Answers: Answers to Commonly Asked "Experienced Internet User" Questions* G.S. Malkin, A.N. Marine, J.K. Reynolds

1208 *Glossary of Networking Terms* O.J. Jacobsen, D.C. Lynch

1213 *Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II* K. McCloghrie, M.T. Rose

1215 *Convention for Defining Traps for Use with the SNMP* M.T. Rose

1228 *SNMP-DPI Simple Network Management Protocol Distributed Program Interface* G.C. Carpenter, B. Wijnen

1229 *Extensions to the Generic-Interface MIB* K. McCloghrie

1230 *IEEE 802.4 Token Bus MIB* K. McCloghrie, R. Fox

1231 *IEEE 802.5 Token Ring MIB* K. McCloghrie, R. Fox, E. Decker

1236 *IP to X.121 Address Mapping for DDN* L. Morales, P. Hasse

1267 *A Border Gateway Protocol 3 (BGP-3)* K. Lougheed, Y. Rekhter

1268 *Application of the Border Gateway Protocol in the Internet* Y. Rekhter, P. Gross

1269 *Definitions of Managed Objects for the Border Gateway Protocol (Version 3)* S. Willis, J. Burruss

1270 *SNMP Communications Services* F. Kastenholz, ed.

1321 *The MD5 Message-Digest Algorithm* R. Rivest

1323 *TCP Extensions for High Performance* V. Jacobson, R. Braden, D. Borman

1325 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* G.S. Malkin, A.N. Marine

1340 *Assigned Numbers* J.K. Reynolds, J.B. Postel

1348 *DNS NSAP RRs* B. Manning

1349 *Type of Service in the Internet Protocol Suite* P. Almquist

1350 *TFTP Protocol* K.R. Sollins

1351 *SNMP Administrative Model* J. Davin, J. Galvin, K. McCloghrie

1352 *SNMP Security Protocols* J. Galvin, K. McCloghrie, J. Davin

1353 *Definitions of Managed Objects for Administration of SNMP Parties* K. McCloghrie, J. Davin, J. Galvin

1354 *IP Forwarding Table MIB* F. Baker

1356 *Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode* A. Malis, D. Robinson, R. Ullmann

1363 *A Proposed Flow Specification* C. Partridge

1372 *Telnet Remote Flow Control Option* D. Borman, C. L. Hedrick

1374 *IP and ARP on HIPPI* J. Renwick, A. Nicholson

1381 *SNMP MIB Extension for X.25 LAPB* D. Throop, F. Baker

1382 *SNMP MIB Extension for the X.25 Packet Layer* D. Throop

1387 *RIP Version 2 Protocol Analysis* G. Malkin

1388 *RIP Version 2—Carrying Additional Information* G. Malkin

1389 *RIP Version 2 MIB Extension* G. Malkin

1390 *Transmission of IP and ARP over FDDI Networks* D. Katz

1393 *Traceroute Using an IP Option* G. Malkin

1397 *Default Route Advertisement In BGP2 And BGP3 Versions of the Border Gateway Protocol* D. Haskin

1398 *Definitions of Managed Objects for the Ethernet-Like Interface Types* F. Kastenholz

1416 *Telnet Authentication Option* D. Borman, ed.

1464 *Using the Domain Name System to Store Arbitrary String Attributes* R. Rosenbaum

1469 *IP Multicast over Token-Ring Local Area Networks* T. Pusateri

1535 *A Security Problem and Proposed Correction With Widely Deployed DNS Software* E. Gavron

1536 *Common DNS Implementation Errors and Suggested Fixes* A. Kumar, J. Postel, C. Neuman, P. Danzig, S.Miller

1537 *Common DNS Data File Configuration Errors* P. Beertema

1540 *IAB Official Protocol Standards* J.B. Postel

1571 *Telnet Environment Option Interoperability Issues* D. Borman

1572 *Telnet Environment Option* S. Alexander

1577 *Classical IP and ARP over ATM* M. Laubach

1583 *OSPF Version 2* J. Moy

1591 *Domain Name System Structure and Delegation* J. Postel

1592 *Simple Network Management Protocol Distributed Protocol Interface Version 2.0* B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters

1594 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* A.N. Marine, J. Reynolds, G.S. Malkin

1695 *Definitions of Managed Objects for ATM Management Version 8.0 Using SMIv2* M. Ahmed, K. Tesink

1706 *DNS NSAP Resource Records* B. Manning, R. Colella

1713 *Tools for DNS debugging* A. Romao

1723 *RIP Version 2—Carrying Additional Information* G. Malkin

1766 *Tags for the Identification of Languages* H. Alvestrand

1794 *DNS Support for Load Balancing* T. Brisco

1832 *XDR: External Data Representation Standard* R. Srinivasan

1850 *OSPF Version 2 Management Information Base* F. Baker, R. Coltun

1876 *A Means for Expressing Location Information in the Domain Name System* C. Davis, P. Vixie, T. Goodwin, I. Dickinson

1886 *DNS Extensions to support IP version 6* S. Thomson, C. Huitema

1901 *Introduction to Community-Based SNMPv2* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1902 *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1903 *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1904 *Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1905 *Protocols Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1906 *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1907 *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1908 *Coexistence between Version 1 and Version 2 of the Internet-Standard Network Management Framework* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1912 *Common DNS Operational and Configuration Errors* D. Barr

1918 *Address Allocation for Private Internets* Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear

1928 *SOCKS Protocol Version 5* M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones

1939 *Post Office Protocol-Version 3* J. Myers, M. Rose

1981 *Path MTU Discovery for IP version 6* J. McCann, S. Deering, J. Mogul

1982 *Serial Number Arithmetic* R. Elz, R. Bush

1995 *Incremental Zone Transfer in DNS* M. Ohta

1996 *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)* P. Vixie

2010 *Operational Criteria for Root Name Servers* B. Manning, P. Vixie

2011 *SNMPv2 Management Information Base for the Internet Protocol Using SMIv2* K. McCloghrie

2012 *SNMPv2 Management Information Base for the Transmission Control Protocol Using SMIv2* K. McCloghrie

2013 *SNMPv2 Management Information Base for the User Datagram Protocol Using SMIv2* K. McCloghrie

2052 *A DNS RR for specifying the location of services (DNS SRV)* A. Gulbrandsen, P. Vixie

2065 *Domain Name System Security Extensions* D. Eastlake, C. Kaufman

2096 *IP Forwarding Table MIB* F. Baker

2104 *HMAC: Keyed-Hashing for Message Authentication* H. Krawczyk, M. Bellare, R. Canetti

2132 *DHCP Options and BOOTP Vendor Extensions* S. Alexander, R. Droms

2133 *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, W. Stevens

2137 *Secure Domain Name System Dynamic Update* D. Eastlake

2163 *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)* C. Allocchio

2168 *Resolution of Uniform Resource Identifiers using the Domain Name System* R. Daniel, M. Mealling

2178 *OSPF Version 2* J. Moy

2181 *Clarifications to the DNS Specification* R. Elz, R. Bush

2205 *Resource ReSerVation Protocol (RSVP) Version 1* R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin

2210 *The Use of RSVP with IETF Integrated Services* J. Wroclawski

2211 *Specification of the Controlled-Load Network Element Service* J. Wroclawski

2212 *Specification of Guaranteed Quality of Service* S. Shenker, C. Partridge, R. Guerin

2215 *General Characterization Parameters for Integrated Service Network Elements* S. Shenker, J. Wroclawski

2219 *Use of DNS Aliases for Network Services* M. Hamilton, R. Wright

2228 *FTP Security Extensions* M. Horowitz, S. Lunt

2230 *Key Exchange Delegation Record for the DNS* R. Atkinson

2233 *The Interfaces Group MIB Using SMIv2* K. McCloghrie, F. Kastenholz

2240 *A Legal Basis for Domain Name Allocation* O. Vaughn

2246 *The TLS Protocol Version 1.0* T. Dierks, C. Allen

2308 *Negative Caching of DNS Queries (DNS NCACHE)* M. Andrews

2317 *Classless IN-ADDR.ARPA delegation* H. Eidnes, G. de Groot, P. Vixie

2320 *Definitions of Managed Objects for Classical IP and ARP over ATM Using SMIv2* M. Greene, J. Luciani, K. White, T. Kuo

2328 *OSPF Version 2* J. Moy

2345 *Domain Names and Company Name Retrieval* J. Klensin, T. Wolf, G. Oglesby

2352 *A Convention for Using Legal Names as Domain Names* O. Vaughn

2355 *TN3270 Enhancements* B. Kelly

2373 *IP Version 6 Addressing Architecture* R. Hinden, M. O'Dell, S. Deering

2374 *An IPv6 Aggregatable Global Unicast Address Format* R. Hinden, M. O'Dell, S. Deering

2375 *IPv6 Multicast Address Assignments* R. Hinden, S. Deering

2389 *Feature negotiation mechanism for the File Transfer Protocol* P. Hethmon, R. Elz

2428 *FTP Extensions for IPv6 and NATs* M. Allman, S. Ostermann, C. Metz

2460 *Internet Protocol, Version 6 (IPv6) S pecification* S. Deering, R. Hinden

2461 *Neighbor Discovery for IP Version 6 (IPv6)* T. Narten, E. Nordmark, W. Simpson

2462 *IPv6 Stateless Address Autoconfiguration* S. Thomson, T. Narten

2464 *Transmission of IPv6 Packets over Ethernet Networks* M. Crawford

2474 *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers* K. Nichols, S. Blake, F. Baker, D. Black

2535 *Domain Name System Security Extensions* D. Eastlake

2539 *Storage of Diffie-Hellman Keys in the Domain Name System (DNS)* D. Eastlake

2553 *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, W. Stevens

2571 *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

2572 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

2573 *SNMP Applications* D. Levi, P. Meyer, B. Stewart

2574 *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

2575 *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

2578 *Structure of Management Information Version 2 (SMIv2)* K. McCloghrie, D. Perkins, J. Schoenwaelder

2640 *Internationalization of the File Transfer Protocol* B. Curtin

2665 *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson

2672 *Non-Terminal DNS Name Redirection* M. Crawford

2710 *Multicast Listener Discovery (MLD) for IPv6*S. Deering, W. Fenner, B. Haberman

2711 *IPv6 Router Alert Option*C. Partridge, A. Jackson

2758 *Definitions of Managed Objects for Service Level Agreements Performance Monitoring* K. White

2845 *Secret Key Transaction Authentication for DNS (TSIG)* P. Vixie, O. Gudmundsson, D. Eastlake, B. Wellington

2874 *DNS Extensions to Support IPv6 Address Aggregation and Renumbering* M. Crawford, C. Huitema

2941 *Telnet Authentication Option* T. Ts'o, ed., J. Altman

2942 *Telnet Authentication: Kerberos Version 5* T. Ts'o

2946 *Telnet Data Encryption Option* T. Ts'o

2952 *Telnet Encryption: DES 64 bit Cipher Feedback* T. Ts'o

2953 *Telnet Encryption: DES 64 bit Output Feedback* T. Ts'o, ed.

3060 *Policy Core Information Model—Version 1 Specification* B. Moore, E. Ellesson, J. Strassner, A. Westerinen

Draft RFCs

Several areas of IPv6 implementation include elements of the following draft RFCs and are subject to change during the RFC review process.

Advanced Sockets API for IPv6
W. Richard Stevens, Matt Thomas, Erik Nordmark, Tatuya Jinmei

Basic Socket Interface Extensions for IPv6
R.E. Gilligan, S. Thomson, J. Bound, J. McCann, W. R. Stevens

Default Address Selection for IPv6
R. Draves

Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification
A. Conta, S. Deering

IP Version 6 Addressing Architecture
R. Hinden, S. Deering

Appendix G. Information APARs

This appendix lists information APARs for IP and SNA documents.

Notes:

1. Information APARs contain updates to previous editions of the manuals listed below. Documents updated for V1R4 are complete except for the updates contained in the information APARs that may be issued after V1R4 documents went to press.
2. Information APARs are predefined for z/OS V1R4 Communications Server and may not contain updates.
3. Information APARs for OS/390 documents are in the document called *OS/390 DOC APAR and PTF ++HOLD Documentation*, which can be found at http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IDDOCMST/CCONTENTS.
4. Information APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation*, which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

Information APARs for IP documents

Table 29 lists information APARs for IP documents.

Table 29. IP information APARs

Title	z/OS CS V1R4	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8
IP API Guide	ii13255	ii12861	ii12371	ii11635
IP CICS Sockets Guide	ii13257	ii12862		ii11626
IP Configuration				ii11620 ii12068 ii12353 ii12649 ii13018
IP Configuration Guide	ii13244	ii12498 ii13087	ii12362 ii12493 ii13006	
IP Configuration Reference	ii13245	ii12499	ii12363 ii12494 ii12712	
IP Diagnosis	ii13249	ii12503	ii12366 ii12495	ii11628
IP Messages Volume 1	ii13250	ii12857 ii13229	ii12367	ii11630 13230
IP Messages Volume 2	ii13251	ii12858	ii12368	ii11631
IP Messages Volume 3	ii13252	ii12859	ii12369 12990	ii11632 ii12883
IP Messages Volume 4	ii13253	ii12860		
IP Migration	ii13242	ii12497	ii12361	ii11618

Table 29. IP information APARs (continued)

Title	z/OS CS V1R4	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8
IP Network and Application Design Guide	ii13243			
IP Network Print Facility		ii12864		ii11627
IP Programmer's Reference	ii13256	ii12505		ii11634
IP and SNA Codes	ii13254	ii12504	ii12370	ii11917
IP User's Guide			ii12365 ii13060	ii11625
IP User's Guide and Commands	ii13247	ii12501	ii12365 ii13060	ii11625
IP System Admin Guide	ii13248	ii12502		
Quick Reference	ii13246	ii12500	ii12364	

Information APARs for SNA documents

Table 30 lists information APARs for SNA documents.

Table 30. SNA information APARs

Title	z/OS CS V1R4	z/OS CS V1R2	CS for OS/390 2.10 and z/OS CS V1R1	CS for OS/390 2.8
Anynet SNA over TCP/IP				ii11922
Anynet Sockets over SNA				ii11921
CSM Guide				
IP and SNA Codes	ii13254	ii12504	ii12370	ii11917
SNA Customization	ii13240	ii12872	ii12388	ii11923
SNA Diagnosis	ii13236	ii12490 ii13034`	ii12389	ii11915
SNA Messages	ii13238	ii12491	ii12382 ii12383	ii11916
SNA Network Implementation Guide	ii13234	ii12487	ii12381	ii11911
SNA Operation	ii13237	ii12489	ii12384	ii11914
SNA Migration	ii13233	ii12486	ii12386	ii11910
SNA Programming	ii13241	ii13033	ii12385	ii11920
Quick Reference	ii13246	ii12500	ii12364	ii11913
SNA Resource Definition Reference	ii13235	ii12488	ii12380 ii12567	ii11912 ii12568
SNA Resource Definition Samples				
SNA Data Areas	ii13239	ii12492	ii12387	ii11617

Other information APARs

Table 31 on page 809 lists information APARs not related to documents.

Table 31. Non-document information APARs

Content	Number
OMPROUTE	ii12026
iQDIO	ii11220
index of recomended maintenace for VTAM	ii11220
CSM for VTAM	ii12657
CSM for TCP/IP	ii12658
AHHC, MPC, and CTC	ii01501
DLUR/DLUS for z/OS V1R2	ii12986
Enterprise Extender	ii12223
Generic resources	ii10986
HPR	ii10953
MNPS	ii10370
Performance	ii11710 ii11711 ii11712

Appendix H. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

IBM may not offer all of the products, services, or features discussed in this document. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O.Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly

tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

This product includes cryptographic software written by Eric Young.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats of unlicensed books and the z/OS Licensed Product Library (LK3T-4307), which contains BookManager and PDF formats of licensed books.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Peer-to-Peer Networking
AFP
AD/Cycle
AIX
AIX/ESA
AnyNet
APL2
AS/400
AT
BookManager
BookMaster
CBPDO
C/370
CICS
CICS/ESA
C/MVS
Common User Access
C Set ++
CT
CUA
DATABASE 2
DatagLANce
DB2
DFSMS
DFSMSdfp
DFSMShsm
DFSMS/MVS
DPI
Domino
DRDA
eNetwork
Enterprise Systems Architecture/370
ESA/390
ESCON
eServer
ES/3090
ES/9000
ES/9370
EtherStreamer
Extended Services
FAA
Micro Channel
MVS
MVS/DFP
MVS/ESA
MVS/SP
MVS/XA
MQ
Natural
NetView
Network Station
Nways
Notes
NTune
NTuneNCP
OfficeVision/MVS
OfficeVision/VM
Open Class
OpenEdition
OS/2
OS/390
OS/400
Parallel Sysplex
Personal System/2
PR/SM
PROFS
PS/2
RACF
Resource Link
Resource Measurement Facility
RETAIN
RFM
RISC System/6000
RMF
RS/6000
S/370
S/390
SAA
SecureWay
Slate
SP
SP2
SQL/DS
System/360

FFST
FFST/2
FFST/MVS
First Failure Support Technology
GDDM
Hardware Configuration Definition
IBM
IBMLink
IBMLINK
IMS
IMS/ESA
InfoPrint
Language Environment
LANStreamer
Library Reader
LPDA
MCS
System/370
System/390
SystemView
Tivoli
TURBOWAYS
UNIX System Services
Virtual Machine/Extended Architecture
VM/ESA
VM/XA
VSE/ESA
VTAM
WebSphere
XT
z/Architecture
z/OS
z/OS.e
zSeries
400
3090
3890

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

DB2 and NetView are registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the U.S., other countries, or both.

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see http://www.intel.com/sites/corporate/tradmarx.htm .

Other company, product, and service names may be trademarks or service marks of others.

Index

Special Characters

/etc/ftp.data 389
/etc/hosts
  accessing HOSTS.SITEINFO 31
/etc/inetd.conf
  adding applications to 719
  configuring for Popper 696
  configuring z/OS UNIX REXECD 708
  definition 709
  setting traces in 719
/etc/osnmpd.data 22
/etc/pagent.conf 562
/etc/protocol 33
/etc/pw.src 23
/etc/resolv.conf
  overview 109
  use of system names in 110
  verifying z/OS UNIX environment with onetstat 148
/etc/services 33, 108, 374, 645, 779
  defining ports for OROUTED 779
  defining ports for Popper 696
  defining ports for RSHD 709
  defining ports for z/OS UNIX REXECD 708
  specifying syslog service 108
/etc/snmpd.boots 632
/etc/snmpd.conf 631
/etc/snmptrap.dest 24
/etc/syslog.conf
  configuring for syslogd 101, 719
  for FTP messages and traces 385
  overview 39
/etc/trapfwd.conf 647
.onslookuprc file, configuring nslookup with 464
'SIOCSVIPA' ioctl 221
'SIOCSVIPA' IOCTL 66

Numerics

328x printer support 315

A

access control
  Fast Response Cache Accelerator 85
  netstat 85
  network 83
  port 82
  stack 81
accessibility features 811
accounting, SMF records
  FTP 40, 393, 396
  PROFILE.TCPIP 113, 129, 150
  syslogd 106
  Telnet 40, 370
active route, configuring
  NCPROUTE 302
  RouteD 791
advertisements, router 162
AF_INET problems 70
alias names 673
anonymous logins, configuring FTP for 400
APPL statement for SNALINK LU0 273
APPL statement for SNALINK LU6.2 277
applications
  configuration files for TCP/IP 18, 26
  planning scenarios for multiple instances 219
applications, functions and protocols
  Character Generator protocol 713
  Discard protocol 713
  Echo protocol 713
  NCP Routing (NCPROUTE) 286
  Network Computing System (NCS) 657
  Network Database System (NDB) 659
  OROUTED Protocol 769
  Portmapper 653, 656, 659
  Remote Execution Protocol Daemon (REXECD) 705, 708
  Remote Printing 649
  Remote Procedure Call (RPC)
    Network Computing System (NCS) 657
    Network Data Base (NDB) 659
    Portmapper 654
  Routing Information Protocol (RIP) 286, 770
  Simple Mail Transfer Protocol (SMTP) 669
  Simple Network Management Protocol (SNMP) 623
  SNALINK LU type 0 269
ARM (automatic restart manager) 38
ARPTO (IPCONFIG ARPTO) 113
AS (autonomous system) 166
  definition 155
ATCCON member of VTAMLST 75
authorization, TCP/IP started task user ID 47
authorization, z/OS UNIX superuser 48
autoconfiguration, stateless 137
AUTOLOG 384
automated takeover, VIPA 216
automatic restart manager (ARM) 38
AUTOMOUNT 390
autonomous system, see also AS 155

B

backing up an MVS host with VIPA 215
banner page 651
Berkeley Internet Name Domain (BIND) 417
BIND (Berkeley Internet Name Domain) 417
BIND 4.9.3 417
BIND 9 417
  DNSSEC 476
  Dynamic update 470
  Incremental zone transfers (IXFR) 470
  IPv6 479
  multiple stack considerations 469
  Split DNS) 471
  TSIG 475

BLKSIZE 390
boot file
  creating 428
  translating 428
BPX.DAEMON facility class 48, 49
BPX.DEFAULT.USER facility class profile 46, 47
BPX.SMF 43, 101, 106
BPXPRMxx
  CINET configuration 64
BPXPRMxx, for defining z/OS UNIX environment 50
BPXPRMxx, role in AF_INET problems 70
BUFNO 390

C

cataloged procedures
  MISCSERV (MISCSERV) 715
  NDBSETUP (NDBSETUP) 660
  PORTC (PORTCPRC) 662
  PORTS (PORTSPRC) 662
  RXSERVE (RXPROC) 705, 708
  SNMPD (SNMPDPRC) 636
  SNMPQE (SNMPPROC) 637
Cisco
  Multi-Node Load Balancer (MNLB) 261, 264
CLAWUSEDOUBLENOP 113
code page IBM-1047, translating to 428, 438, 439
commands
  MODIFY (MVS)
    Remote Execution server 707
    SNALINK LU0 275
  START (MVS) 76
COMMONSEARCH 13, 16
Communications Server for z/OS, online information xxi
component trace, customizing 71
CONDDISP 390
configuration data sets
  ETCRPC 654
  HOSTS 145
  NPSIDATE 280
  NPSIGATE 280
  SAMPPROF 110
  SMTPCONF 681
  SMTPNOTE 671
  VTAMLST
    in SNALINK LU0 273
    in SNALINK LU6.2 277
    in X.25 NPSI 283
  X25CONF 280
configuring
  data set naming conventions 19
  dynamic VIPA 218
  files for TCP/IP applications 26
  files for the TCP/IP stack 25
  OROUTED 769
  resolver environment variables 29
  searching for data sets 18
  SNMP for z/OS UNIX 623
  SNTPD 701
  TFTP server 408
configuring *(continued)*
  TIMED 699
  verifying for dynamic VIPAs 241
configuring host resolvers, onslookup considerations 457
Configuring the z/OS UNIX Telnet server 374
connection optimization
  configuring a sysplex domain
    choosing sysplex name 492
    configuring client applications 494
    configuring for WLM registration 491, 495
    configuring name servers 493
    configuring WLM in goal mode 495
    identifying server applications 490
    updating parent name server 492
  configuring a sysplex domain for
    identifying name servers 492
  overview 482
control characters, TCP/IP messages 77
conversion characters, TCP/IP messages 77
cryptography 87
CTRACE keyword 71
customizing
  SMTP mail headers 672
customizing TCP/IP messages 76

D

data sets
  dynamic allocation 19
  naming conventions 19
  overview 11
  search order for 18
DATACLASS 390, 391
DATAGRAMFWD (IPCONFIG DATAGRAMFWD) 113, 240, 778
DB2 396, 406
DB2 connection authorization exit routine 660
DB2 SQL
  in FTP server 406
  in NDB 659
DB2PLAN 396
DCBDSN 390
DD cards 25, 26
DDNS (dynamic domain name services) 497
default route, configuring
  NCPROUTE 303
  RouteD 792
DEFAULTIPNODES 14, 16
DEFAULTTCPIPDATA 13, 16
DHCP (dynamic host configuration protocol) 497
Differentiated Services (DS)
  Policies 565
DIRECTORY 390
disability, physical 811
distributed VIPA 209
DNS (Domain Name System)
  authoritative servers 420
  caching-only servers 421
  definitions 417
  dynamic update 426, 470

DNS (Domain Name System) *(continued)*
forward data files 433
forwarders 421
Logging, for BIND 9 442
master name servers 420
overview 417
problem diagnosis 467
Queries 425
reverse data files 433
slave name servers 421
slave name servers, configuring 450
SOURCEVIPA 466, 468
stealth server 421
synthetic IPv6 responses 480
SYSPLEXROUTING 491
translating boot files 428
translating data files 438, 439
Zone transfers 425
DNS, online information xxii
DNS, security 98
DNS/WLM 261
DNSSEC 476
documents, licensed xxii
Domain Name Resolution, SMTP 686
Domain Name System, see DNS 417
DSN3SATH 660
DSNLOAD 408
duplicate address detection 137
DVIPA takeover
overview 229
using IPSec with 231
DVIPSEC 228
dynamic domain name services (DDNS) 497
dynamic host configuration protocol (DHCP) 497
dynamic IP 496
dynamic routes
definition 155
dynamic routing
IPv6 205
using OMPROUTE 166
versus static routing 157
dynamic VIPA
256 limit 215
configuration 218, 240, 241
considerations 239
DNS considerations 462
MODDVIPA utility 222
multiple application-instance scenario 218
overview 209
relationship to UDP 239
resolving conflicts 232
routing protocols 247
unique application-instance scenario 218, 219
use with OMPROUTE 170, 185
verifying configuration using NETSTAT 244
verifying in a sysplex 241
within subnets 239
DYNAMICXCF 252
DYNAMICXCF (IPCONFIG DYNAMICXCF) 112, 114, 127, 226

E

EGP (exterior gateway protocol) 288
definition 155
Enterprise Extender 101
OROUTED and 792
overview 65
VIPA considerations 212, 214
entry point name incorrect 70
environment variables
for overriding default search order 18
FTP server and 388
OMPROUTE use of 174
OROUTED and 779, 789
passing to syslogd process 106
resolver configuration files and 29
REXECD and 709
environment, NCPROUTE 286
ETC.GATEWAY 780
ETC.IPNODES 143, 146
ETC.SERVICES
FTP and 385
NCPROUTE 293
OROUTED and 779
RouteD 779
Express Logon Feature (ELF) 96
overview 749
exterior gateway protocol (EGP) 288
definition 155
external gateway 287, 773
external route, configuring 773
NCPROUTE 302
RouteD 790
EZACFSM1 37
EZASMF76 43
EZASMF77 44
EZAZSSI 73
EZBDVIPA 229, 232
EZBRECNF 16
EZBREPRC 15

F

fast path for socket applications 52
Fast Response Cache Accelerator access control 85
fault tolerance, interface layer for LANs 137
filters, input/output, for RIP 290, 772
FIREWALL (IGNOREREDIRECTS FIREWALL) 113
FTCHKCMD 397
FTCHKIP 396
FTCHKJES 398
FTCHKPWD 397
FTP
/etc/syslog.conf 385
accounting 40, 393
anonymous 400, 405, 414
APPEND 393
AUTOLOG PORT KEEPALIVE 384
cataloged procedure 385, 406
CCXLATE 388
configuration statements, TCP/IP 384

FTP *(continued)*
data translation 393
DB2 406
DELETE 393
ENVAR 388
environment variables for FTP server 388
FTCHKCMD 397
FTCHKIP 396
FTCHKJES 398
FTCHKPWD 397
FTP.DATA data set 389
FTPOSTPR 398
FTPSMFEX 396
JES 399
RACF considerations 386
RENAME 393
RETRIEVE 393
security considerations 386
SMF configuration 393
specifying attributes for new MVS data sets 391
STORE 394
STORE UNIQUE 394
SURROGATE 400
TCPIP.DATA 389
translation of data 393
updating the FTP cataloged procedure 385
user exit 396
XLATE 388
FTP.DATA 389
(FILETYPE=JES) 394
(FILETYPE=SEQ) 394
(FILETYPE=SQL) 394
ANONYMOUSHFSINFO 405
ANONYMOUSLOGINMSG 405
ANONYMOUSMVSINFO 405
ASATRANS 393
AUTOMOUNT 390
BANNER 405
BLKSIZE 390, 391
BLOCKSIZE 390
BUFNO 390
CONDDISP 390
CTRLCONN 393
data set attributes 390
DATACLASS 390, 391
DB2 396
DB2PLAN 396
DCBDSN 390, 391
DIRECTORY 390, 391, 392
dynamic allocation 391
ENCODING 393
EXTENSIONS UTF8 393
HSFINFO 405
JESINTERFACELEVEL 396
JESINTERFACELevel=2 399
JESLRECL 396
JESPUTGETTO 396
JESRECFM 396
LOGINMSG 405
LRECL 390, 391, 392
MBDATACONN 393
FTP.DATA *(continued)*
MGMTCLASS 390, 391
MIGRATEVOL 390
MVSINFO 405
PORTCOMMAND 387
PORTCOMMANDIPADDR 387
PORTCOMMANDPORT 387
PRIMARY 390, 391, 392
RECFM 390, 392
RETPD 390, 392
SBDATACONN 393
SBSUB 393
SBSUBCHAR 393
search order 389
SECONDARY 390, 391, 392
SMFAPPE 393
SMFDEL 393
SMFEXIT 393
SMFJES 393
SMFLOGN 393
SMFREN 393
SMFRETR 393
SMFSQL 393
SMFSTOR 393
SMS 392
SPACETYPE 390, 391
SPREAD 396
SQLCOL 396
STORCLASS 390, 391, 392
UCOUNT 390, 391
UCSHOSTCS 393
UCSSUB 393
UCSTRUNCT 393
UMASK 391
UNITNAME 391, 392
VCOUNT 391
VOLUME 391, 392
XLATE 393
FTPD 385
FTPOEBIND 406
FTPOSTPR 398
FTPSMFEX 396
FTPSMFEX user exit 396

G

gateway
Interior Gateway Protocol (IGP) 770
gateway route table name 294
GATEWAY statement 160
configuring static routes 292, 304
GATEWAY_PDS statement 299
gateways
active 791
active routes 289, 302
data set (NCPROUTE) 300, 303
default routes 303
enabling as DHCP relay agents 501
external routes 288
file configuration for OROUTED 779
NCPROUTE 287, 290

gateways *(continued)*
passive routes 287
resolving names of 152
SMTP 679
TCP-to-NJE mail 681, 683
gateways data set
NCPROUTE 300
RouteD 779
gateways file or data set 780
generic stack affinity 55
GLOBALIPNODES 14, 16
GLOBALTCPIPDATA 12, 16
gskkyman utility 726

H

HCD, using 757
HFS (Hierarchical File System)
concepts 11
security considerations 49
high-level qualifier (HLQ) 19
hints (root server) file
definition 439
HiperSockets
concepts 130
iQDIO 130
HiperSockets Accelerator
efficient routing with 135
HLQ (high-level qualifier) 19
HOMETEST 152
HOSTALIASES 29
HOSTS.ADDRINFO
generating from HOSTS.LOCAL 144
HOSTS.LOCAL 143
HOSTS.SITEINFO
generating from HOSTS.LOCAL 144
verifying 152

I

IBM Software Support Center, contacting xxiii
IDS 99, 595
Defining Policies Using LDAP 603
IEFSSNxx member 671
IGNOREREDIRECTS
FIREWALL 113
IGNOREREDIRECTS (IPCONFIG IGNOREREDIRECTS) 159, 778
IGP (interior gateway protocol), definition 156
IKE (Internet Key Exchange) 91
IKJTSOxx member 672
ImageServer statement 510
in-addr.arpa domain, definition 418
inetd configuration file, setting up 719
inetd listener program 48
information APARs for IP-related documents 807
information APARs for non- document information 808
information APARs for SNA-related documents 808
initialization failure 69, 70
initializing, NCPROUTE 287
input/output filters, RIP 290, 772
installing z/OS CS 68
instances of TCPIP, considerations for multiple 54
interface takeover 137
interface-layer fault-tolerance for LANs 137
interior gateway protocol (IGP), definition 156
Internet Key Exchange (IKE) 91
Internet, finding z/OS information online xxi
InterNetwork Information Center (InterNIC) 418
InterNIC (InterNetwork Information Center) 418
Intrusion Detection Services 99, 540
Intrusion Detection Services (IDS) 595
IDS Policy 547
IP addressing, virtual 209
IPCONFIG
ARPTO 113
DATAGRAMFWD 113, 240, 778
DYNAMICXCF 112, 114, 127, 226
IGNOREREDIRECTS 159, 778
MULTIPATH 113, 164, 167
PATHMTUDISC 114, 159
SOURCEVIPA 113, 128, 213, 466, 468, 778
SYSPLEXROUTING 114, 240, 491
VARSUBNETTING 113, 778
IPSec, security 89
IPv6
autoconfiguration, stateless 137
BPXPRMxx, sample definitions 50
configuring static VIPAs 213
defining TCP/IP as UNIX System Services PFS 50
DNS lookups 480
duplicate address detection 137
dynamic routing 205
inetd configuration file, setting up 719
router advertisements 162
stack functions supported 3
static routing 161
static versus dynamic routing 157
iQDIO 259
concepts 130
IUCV/VMCF 75

J

JES 399
JESINTERFACELEVEL 396
JESLRECL 396
JESPUTGETTO 396
JESRECFM 396

K

Kerberos, security 97
keyboard 811

L

LDAP server 548
Object classes 548
Schema definition 555
license, patent, and copyright information 813
licensed documents xxii

load libraries, protecting with RACF 50
local host table 143
LOCALDOMAIN environment variable, configuring onslookup with 464
log files, offloading 106
loopback file
  definition 441
LPD 649
  banner page 651
  configuration data set 651
  LPDDATA 650
  LPDPRFX 650
  PROFILE.TCPIP changes 649
  tracing 650
LRECL 390
LU assignments - objects, client identifiers, mapping statements 325
LU0, see SNALINK LU0 269
LU6.2, see SNALINK LU6.2 276

M

MAKESITE 145
MD5
  and OSPF 175
messages data sets 76
messages, logging of 39
messages, TCP/IP
  rules for customizing 77
MGMTCLASS 390, 391
MIBS.DATA 641
middle-level qualifier (MLQ) 19
MIGRATEVOL 390
MISC server 713
MLQ (middle-level qualifier) 19
MNLB, Cisco 261, 264
MODDVIPA utility 222
MODDVIPA, defining RACF profile for 223
MODIFY command
  Remote Execution server 707
  SNALINK LU0 275
msys for Setup 6
Multi-Node Load Balancer (MNLB), Cisco 261, 264
MULTIPATH (IPCONFIG MULTIPATH) 113, 164, 167
multiple application-instance scenario 218
multiple copies of TCP/IP 54
multiple stacks
  AUTOLOG 142
  BPXPRMxx 64
  CINET PFS 54
  generic versus specific affinity 55
  OROUTED considerations 788
  OSA/SF considerations 645
  OSPF and RIP considerations 169
  overview 54
  port management 55
  selecting a stack 60
  SMF accounting 43
  socket application programs 60
  TCPIP.DATA 61, 109
  VIPA considerations 211, 216
MVS
  accounting 40
  automatic restart manager (ARM) 38
  component trace 71
  failure management 239
  logging system messages 39
  SERVAUTH 44
  system symbols 37, 111
MX records 686

N

name resolution
  HOMETEST command to verify 153
  in a sysplex domain 483
  iterative resolution 419
  SMTP domain 686
  TESTSITE command to verify 152
  using HOSTS.LOCAL data set 143
  VIPA host 213
name servers
  authoritative 419
  caching-only, definition 421
  configuring master and caching-only 427
  for VIPA host-name resolution 213
  forwarder, definition 421, 426
  master, definition 420
  slave, definition 421
  SMTP configuration for 686
  Stealth, definition 421
named daemon 448
naming conventions, dynamically allocated data sets 20
NCP host interface 296
NCP IP router statements 297
NCPROUTE
  AUTOLOG 291
  BSDROUTINGPARMS 292
  building the NCPROUTE profile 298
  cataloged procedure 293
  configuration examples 303
  configuring 290
    active route 302
    client NCP 294
    default route 303
    external route 302
    GATEWAYS data set 300
    passive route 301
  DD statement for external message data set 76
  defining for TCP/IP 272
  DEVICE 293
  ETC.SERVICES 293
  filters 290
  filters, input/output 290
  gateways 287
  gateways data set 300
  HOME 292
  interaction with VIPA 113
  LINK 293
  NCP 294
  operation 287

NCPROUTE *(continued)*
overview 285, 286
PORT 291
profile data set 298
RIP 286
RIP advertising rules 288
RIP, external 288
RIP, passive 287
server requirements 287
SNMP 286
specifying configuration statements 291
updating ETC.SERVICES 293
use with OMPROUTE 161
VTAM definitions 292
NCS interface
configuration 657
LLBD cataloged procedure 658
NRGLBD cataloged procedure 658
specifying statements in PROFILE.TCPIP 658
NCST (NCP Connectionless SNA Transport) 295
NDB (network database) system
DSN3SATH 661
multiple PORTC procedures 662
NDB client for different platforms 662
NDBSETUP 660
PORTC 662
portmap requirement 659
PORTS 662
starting NDB 668
NETSTAT 71
netstat access control 85
NetView 623, 639
network access control 83
network connectivity, SNA network 269
Network DataBase System (NDB) 659
network file system, see also NFS 653
NFS (network file system)
PORTMAP address space 653
NJE
mail gateway 681
NOCOMMONSEARCH 13, 16
NPSI, see X.25 278
nslookup command
option alternatives 464
nslookup command, overview 463

O

offloading log files 106
OMPROUTE
autolog considerations 172
cataloged procedure 172
configuring 160, 171
displaying information 194
interaction with service policy 170
interaction with VIPA 113, 170, 211
migrating to, from OROUTED 165
multiple stack considerations 169
overview 165, 166, 169
parameters 177
ROUTESA_CONFIG 628
OMPROUTE *(continued)*
run-time environment 168
sample configuration files 202
SNMP subagent 642
starting 176
stopping 178
supported protocols 166
use with NCPROUTE 292
verification of configuration and state 194
OMPROUTE_DEBUG_FILE 175
OMPROUTE_DEBUG_FILE_CONTROL 175
OMPROUTE_FILE 174
OMPROUTE_OPTIONS 174
OMVS RACF segment 45, 47, 70, 71
onslookup command
command line mode 464
interactive mode 464
overview 463
onslookup considerations, configuring host resolvers 457
open shortest path first, see also OSPF 156
Open Systems Adapter (OSA)
with ARP offload 129
with Cisco router 264
with SNMP 642
OPTIONS statement
use with NCPROUTE 301
use with OROUTED 782
OROUTED 769
cataloged procedure 779
command-line parameters 786
configuration examples 789
configuring 769
configuring the OROUTED Server 769
configuring with HiperSockets 796
configuring with OSA-Express in QDIO mode 795
DATAGRAMFWD 778
filters, input/output 772
gateways data set 779
IGNOREREDIRECTS 159, 778
interaction with NCPROUTE 285
interaction with VIPA 113
migrating from, to OMPROUTE 165
overview 165
PATHMTUDISC 159
SOURCEVIPA 778
starting 788
statement options 782
understanding OROUTED 769
VARSUBNETTING 778
OROUTED, understanding 769
osnmp command, configuring 640
OSNMPD, configuring 627
OSNMPD.CONF, search order for 22
OSNMPD.DATA, search order for 22
OSPF (open shortest path first)
configuring authentication 175
configuring OSPF and RIP 179
definition 156
overview 166
sample configuration files 202

OSPF (open shortest path first) *(continued)*
security 98
otelnetd 378

P

PAGENT.CONF 562
parameter, Subnet_mask 182
parameters, LPD server cataloged procedure
DIAG 650
LPDDATA 650
LPDPRFX 650
TRACE 650
TYPE 650
VERSION 650
parameters, Miscellaneous server
CHARGEN 716
DEbug 716
DISCARD 716
ECHO 716
TRACE 716
parameters, OROUTED cataloged procedure
-dp 786
-g 786
-q 787
-s 786
-sd 787
-sdv 787
-st 787
-sv 787
-svd 787
-t 787
-t-t 787
-t-t-t 787
-t-t-t-t 787
parameters, OROUTED gateways data set
active 780
block, options 783
external 781
forward 783
forward.cond 783
host 780
interface, options 783
interface.poll.interval, options 782
interface.scan.interval, options 782
metric, options 781
net 780
passive, options 781, 783
supply off, options 784
parameters, SMTP statements
DEBUG, SMSG 681
EXPIRE, SMSG 681
HELP, SMSG 681
NODEBUG, SMSG 681
NOTRACE, SMSG 681
QUEUES, SMSG 681
SHUTDOWN, SMSG 681
STATS, SMSG 681
TRACE, SMSG 681
passive gateway 287, 773
passive route, configuring
NCPROUTE 301
OROUTED 789
path length 52
PATHMTUDISC (IPCONFIG PATHMTUDISC) 114, 159
performance considerations 52
performance monitoring, SLA subagent 588
PFS (physical file system) 50, 54
physical file system (PFS) 54
point-to-point link, configuring (OROUTED) 792
Policies
Attack 598
defining using LDAP 574
Differentiated Services (DS) 565
DS 571
IDS Attack 608
IDS Scan 605
IDS TR 615
IDS TR TCP 601
IDS TR TCP, using Policy Agent 603
IDS TR UDP 602
IDS, using LDAP 603
in Policy Agent configuration file 570
Integrated Services (RSVP) 567
RSVP 572
RSVP in LDAP 579
Scan 595
Sysplex Distributor 573
Sysplex Distributor (SD) 567
Sysplex Distributor in LDAP 580
Traffic Regulation (TR) 601
Policy Agent 539
and LDAP objects 554
Configuration file 570
Configuring 557
sample files 541
sample LDAP objects, using 556
Starting and stopping 562
popper 689
port access control 82
port management
multiple stacks 55
port ownership, specifying 431
PORTMAP
cataloged procedure 654
configuring 653
ETC.RPC 654
required by NFS 653
starting 656
PORTMAP address space
configuring 653, 656
starting PORTMAP 656, 657
updating the PORTMAP cataloged procedure 654, 657
PortMapper
cataloged procedure 657
starting 657
PortMapper, z/OS UNIX
configuring 656
PORTRANGE
TCP/IP profile statements 143

PRIMARY 390
printer support, 328x 315
printf function 77
problem diagnosis, DNS
  checking syslog messages 467
  using name server signals 467
  using nslookup 468
procedures, TCP/IP
  MISCSERV (MISCSERV) 715
  NDBSETUP (NDBSETUP) 660
  PORTC (PORTCPRC) 662
  PORTS (PORTSPRC) 662
  RXSERVE (RXPROC) 705, 708
  SNMPD (SNMPDPRC) 636
  SNMPQE (SNMPPROC) 637
PROFILE.TCPIP
  ARPAGE 113
  ARPTO 113
  AUTOLOG 141
  BEGINROUTES 129
  BSDROUTINGPARMS 114
  changes needed for FTP 384
  CLAWUSEDOUBLENOP 113
  DATAGRAMFWD 113
  DATASETPREFIX 19
  DELAYACKS 114
  DEVICE 126
  DYNAMICXCF 114
  ECSALIMIT 113
  FINWAIT2TIME 114
  FIREWALL 113
  GLOBALCONFIG 113
  HOME 128
  IGNOREREDIRECT 113
  INTERFACE 128
  IPCONFIG 113
  IPCONFIG6 114
  LINK 126
  MULTIPATH 113
  MVS system symbols 37
  netstat 149
  NOUDPCHKSUM 115
  PATHMTUDISCOVERY 114
  physical characteristics, setting up 115
  PING 151
  POOLLIMIT 113
  PORT 60, 142, 385, 645
  PRIMARYINTERFACE 128
  REASSEMBLYTIMEOUT 114
  reserved port number definitions, setting up 139
  RESTRICTLOWPORTS 114, 115
  SACONFIG 642, 645
  sample 115
  search order 25, 110
  SENDGARBAGE 114
  SOMAXCON 114
  SOURCEVIPA 113, 128
  STOPONCLAWERROR 114
  SYSPLEXROUTING 114
  TCP/IP operating characteristics, setting up 112
  TCPCONFIG 114, 385
PROFILE.TCPIP *(continued)*
  TCPIPSTATISTICS 113
  TCPMAXRCVBUFRSIZE 114, 385
  TCPRCVBUFRSIZE 114
  TCPSENDBFRSIZE 114
  TCPTIMESTAMP 114
  TRACERTE 151
  TRANSLATE 129
  UDPCONFIG 115
  UDPQUEUELIMIT 115
  UDPRCVBUFRSIZE 115
  UDPSENDBFRSIZE 115
  VARSUBNETTING 113
  verifying your configuration 148
PROFILE.TCPIP, specifying configuration statements
  EZAFTSRV 385
  NCPROUTE 291
  OROUTED 777
  PORTMAP 653, 656
  SMTP 670
  SNALINK 270
  TCPIP 110
  X.25 NPSI 279
program control 49
program directory 68
protocol suite 39
pwtokey 633

Q

QoS, see Quality of Service (QoS) 565
Quality of Service (QoS)
  and Policy Agent 567
  QoS Policy 546

R

RACF
  Common Keyring support 726
  considerations for FTP server 386
  considerations for REXEC server 706
RACF (Resource Access Control Facility) 45, 47, 49, 70, 71
  authorizing sources 45
  FTPD 386
  port access control 82
  resource protection 81
  REXEC access to MVS 706
  stack access control 81
  starting OMPROUTE 174
  starting OROUTED 785
  user access control 81
RACF profile, defining for MODDVIPA 223
REASSEMBLYTIMEOUT 114
RECFM 390
registration, WLM
  overview 483
  server applications
    customized applications 495
    TCP/IP 491
    TN3270 491

remote hosts, accessing using Telnet 305
RESOLVE_VIA_LOOKUP 144
resolver configuration files
  for host names outside local area 143
  MVS versus z/OS UNIX resolver 27
  overview 27
  search order 27
  setting environment variables 29
  TCPIP.DATA 109
  use with OMPROUTE 172
  use with OROUTED 778
resolver setup file statements
  COMMONSEARCH 13, 16
  DEFAULTIPNODES 14, 16
  DEFAULTTCPIPDATA 13, 16
  GLOBALIPNODES 14, 16
  GLOBALTCPIPDATA 12, 16
  NOCOMMONSEARCH 13, 16
resolver setup file, sample 16
resolver start procedure, sample 15
RESOLVER_CONFIG
  overview 29
  pointing to TCPIP.DATA 109
  setting the value of 29
  use by OMPROUTE 174
  use with OROUTED 789
  when running multiple TCP/IP stacks 63
RESOLVER_IPNODES 29
resolvers
  and BIND 9 DNS 14
  and dig 14
  and IBM APIs 14
  and nslookup 14
  and nsupdate 14
  and SMTP 14
  customization 15
  managing the resolver address space 18
  setting up 14
  starting and stopping 18
  understanding resolvers 12
resolvers, configuring host
  name server considerations 457
  nslookup considerations 466
resolving conflicts, VIPA 232
RESOPROC 15
Resource Access Control Facility, see also RACF 45
Resource Access Control Facility, z/OS UNIX security
  and, see also RACF 45
RESSETUP 16
RETPD 390
REXECD 48
  cataloged procedure 707
  configuring PROFILE.TCPIP 705
  security considerations 706
  UNIX 705
  user exits 707
  userid.RHOSTS.DATA 706
REXECD, z/OS UNIX
  configuring inetd 719
  considerations in CINET environment 58
  HFS files 708
REXECD, z/OS UNIX *(continued)*
  installation 708
RFC (request for comment)
  list of 797
RFC (request for comments)
  accessing online xxi
RIP (Routing Information Protocol)
  configuring 179
  definition 156, 167
  external routes and NCPROUTE 288
  input/output filters 290
  interaction with NCPROUTE 285, 286
  interaction with VIPA 249
  OROUTED support of 774
  overview 770
  passive routes and NCPROUTE 287
  reserving RIP UDP port for OMPROUTE 172
  route advertising rules 288
  sample configuration files 202
RIP input/output filters 290, 772
RIP_RECEIVE_CONTROL statement 298
RIP_SUPPLY_CONTROL statement 298
RIP2_AUTHENTICATION_KEY statement 298
root file system 12
router advertisements 162
router, definition 156
routing
  daemons 156, 165
  definition 156
  dynamic VIPAs 247
  IGNOREREDIRECTS 159
  IPv6 dynamic 205
  IPv6 static 161
  MULTIPATH 164, 167
  network design considerations 193
  PATHMTUDISC 159
  Routing Information Protocol (RIP) 286
  routing information tables 294
  routing table 286
  SOURCEVIPA 213
  static versus dynamic 157
  verification of 205
Routing Information Protocol (RIP) 770
Routing Information Protocol, see also RIP 156
routing table 770
RPCINFO 654
RSHD 48
RSHD, z/OS UNIX
  configuring inetd 719
  considerations in CINET environment 58
  HFS files 709
  installation exit 710
RSVP 583
  Configuring 584
  Policies 567
  Starting and stopping 584
RSVP Agent 540

S

sample NCP IP router statements 297

search order
configuration files 18
DATASETPREFIX value 19
ETC.IPNODES 32, 36
ETC.PROTO 21, 33, 36
ETC.SERVICES 21, 33, 37
FTP.DATA 21, 389
GATEWAYS configuration data set or file 779
high-level qualifier (HLQ) 19
HOSTS.ADDRINFO 32, 35
HOSTS.SITEINFO 31, 35
LPD configuration file 650
MIBS.DATA 641
middle-level qualifier (MLQ) 19
OROUTED configuration data set or file 774
OSNMPD.CONF 22
OSNMPD.DATA 22
overview 18
PAGENT.CONF 22
PROFILE.TCP/IP 25
PROFILE.TCPIP 22
PW.SRC 23
resolver configuration files 27
RSVPD.CONF 23
SERVICES data set or file 779
SNMPD.BOOTS 23
SNMPD.CONF 23
SNMPTRAP.DEST 24
STANDARD.TCPXLBIN 31, 34
TCPIP.DATA 26
TRAPFWD.CONF 24
with DD cards in TCP/IP startup procedure 25
without DD cards in TCP/IP startup procedure 26
SECONDARY 390
Secure Socket Layer, see SSL 721
security
application 79
event reporting 99
Express Logon Feature (ELF) 96
FTP server 386
IPSec 89
overview 79
principals 87
protecting data in the network 87
protocols 89
RACF 45
resource protection 81
SSL and TLS 93
security, z/OS UNIX considerations 45, 47, 48, 49
sendmail, z/OS UNIX 689
SERVAUTH 44, 139, 323
MVS considerations 44
resource protection 81
restricting access to port numbers by applications 44
restricting access to TSO and UNIX shell Netstat command 45
setting up 143
server requirements, NCPROUTE 287
ServerType statement 510
service policy agent
SNMP 628
SNMP subagent 642
SESSLIM parameter, VTAMLST 76
setup file statements, resolver
COMMONSEARCH 13, 16
DEFAULTIPNODES 14, 16
DEFAULTTCPIPDATA 13, 16
GLOBALIPNODES 14, 16
GLOBALTCPIPDATA 12, 16
NOCOMMONSEARCH 13, 16
setup file, sample resolver 16
Setup, z/OS msys for 6
shortcut keys 811
Simple Network Management Protocol, see also SNMP 623
Simple Network Time Protocol (SNTP) 701
SLA subagent 585
performance monitoring 588
traps generated by 630
SMF (System Management Facility)
record type 118 43
record type 119 44
records for FTP 40, 393
records for Telnet 40
see also, accounting 40
user exit for FTP server 396
SMS (Storage Management System) 392
SMTP
configuring 669
exit to filter unwanted mail 687
SMTP headers, customizing 672
SMTP.RULES data set 673
SMTP.SECTABLE data set 683
SNA network connectivity 269
SNALINK environment 269
SNALINK LU0 269
AUTOLOG 274
BEGINROUTES 270
BSDROUTINGPARMS 270
cataloged procedure 273
configuring 270
connections 275
definitions 272
DEVICE 270
dynamic routing 269
GATEWAY 270
HOME 270
LINK 270
MODIFY 275
NETSTAT DEVLINKS 275
PROFILE.TCPIP 270
sample console 274
starting 273
stopping 273
verifying 275
VTAM definitions 273
SNALINK LU6.2 276
cataloged procedure 277
configuration data set 278
configuring 276

SNALINK LU6.2 *(continued)*
DEVICE 277
LINK 277
VTAM definitions 277
SNMP
agents and subagents 627
configuring 623
overview 623
updating the SNMPD cataloged procedure 637
SNMP (Simple Network Management Protocol)
agents and subagents 642
community names 629
community-based security 629, 637
configuring for NCPROUTE 298
configuring for z/OS UNIX 623
creating user keys 633
DD statement for external message data set 76
enabling traps for SLA subagent 589
Enterprise-Specific variables 639
MIBDESC.DATA 637
multiple SNMPv3 agents in same MVS image 23, 633
NetView 637
OSA 642
OSNMP 640
OSNMPD, starting 635
OSNMPD.DATA 22
overview 623
port specification 628
PW.SRC 23
pwtokey 633
security 625
SNMPD.BOOTS 632
SNMPD.CONF 631
SNMPTRAP.DEST 24, 630
SNMPv1, SNMPv2C, SNMPv3 625
TCP/IP profile statements 143, 627, 645
textual names 641
trap forwarding 646
TRAPFWD.CONF 647
user-based security 631
SNMP SLA Subagent 540
SNMP_AGENT statement 299
SNMP_COMMUNITY statement 299
SNMPIUCV module 639
SNMPv3, security 98
SNTPD daemon 701
socket applications (z/OS UNIX), support for fast path 52
socket applications use of z/OS UNIX 45
SOURCEVIPA (IPCONFIG SOURCEVIPA) 113, 128, 213, 466, 468, 778
SPACETYPE 390
specific stack affinity 55
specifying configuration statements in PROFILE.TCPIP 291
splitting network traffic 793
SPREAD 396
SQL 406
SQL usage
in FTP server 406
SQL usage *(continued)*
in NDB 659
SQLCOL 396
SSL
for DCAS 721
for FTP server 721
for Telnet server 721
overview 721
SSL, security 93
stack access control 81
stack affinity, specifying 431
stack communications, z/OS UNIX to TCP/IP 52
stack functions supported, IPv6 3
STANDARD.TCPXLBIN 31, 34
START command 76
start procedure, sample resolver 15
started task 46, 47
static routes
definition 156
static routing
configuration examples 162
IPv6 161
using with OMPROUTE 160
versus dynamic routing 157
static routing, IPv4 158
Storage Management System (SMS) 392
STORCLASS 390, 391
subagent, SLA performance monitoring 588
Subnet_mask parameter 182
subnets, dynamic VIPAs within 239
superuser authorization 48
SURROGATE, in anonymous logins 401
symbols, MVS system 37
SYSFTPD 389
syslog file, creating 458
syslogd
command format 105
configuring 101
diagnosing configuration problems 108
exit values 106
for z/OS UNIX applications 107
overview 39, 108
stopping 106
sysplex
configuring for connection optimization 490
failure management 239
name resolution in 483
overview 482
sysplex wide dynamic source VIPAs for TCP connections 226
Sysplex Wide Security Associations (SWSA) 228
SYSPLEXPORTS 227
TCP/IP in a 251
workload balancing 260
Sysplex Distributor 209, 546, 573, 580, 587
configuring distributed DVIPAs 224
DATAGRAMFWD 240
DNS considerations 462
DYNAMICXCF 112
policy interactions 261
SYSPLEXROUTING 240

Sysplex Distributor *(continued)*
using IPSec with 231
with Cisco routers 261
with SWSA 230
workload verification 246
Sysplex Distributor (SD)
Policies 567
sysplex wide dynamic source VIPAs for TCP connections 226
Sysplex Wide Security Associations (SWSA)
DVIPA takeover 229
EZBDVIPA 229, 232
IPSec with DVIPAs and Sysplex Distributor 231
overview 228
Sysplex Distributor 230
SYSPLEXPORTS 227
SYSPLEXROUTING (IPCONFIG SYSPLEXROUTING) 114, 240, 491
SYSTCPD DD
fork() considerations 30, 109
search order for TCPIP.DATA 109
SNALINK LU6.2 cataloged procedure 277
System Management Facility, see also SMF 43
system symbols, MVS 37

T

takeover, DVIPA 229
tasks
migrating an existing DNS configuration to BIND 4.9.3 dynamic IP
steps for 498
starting SNTPD as a procedure
steps for 702
starting SNTPD from the z/OS shell
steps for 701
TCP/IP
application configuration files 26
changing configuration information 111
configuration data sets 21
configuration files for the stack, search order 25
customizing messages 76
installation, planning for 67
multiple instances 54
online information xxi
protocol specifications 797
resolver configuration files 27
search order and configuration files for the stack 18
stack configuration files, search order 25
starting the address space 76
startup DD cards 25
sysplex considerations 251
TCPIP.DATA
/etc/resolve.conf 109
characteristics 109
configuring for FTP 389
configuring nslookup 464
creating 109
DATASETPREFIX 19, 63
finding with SYS1.TCPPARMS 20
multiple stacks 61
TCPIP.DATA *(continued)*
MVS system symbols 37
overview 109
search order 26
setting default with DEFAULTTCPIPDATA 13
Specifying a default local host file with DEFAULTIPNODES 14
Specifying a global local host file with GLOBALIPNODES 14
specifying global resolver settings with GLOBALTCPIPDATA 12
Specifying local host file search order with COMMONSEARCH 13
Specifying local host file search order with NOCOMMONSEARCH 13
syntax 110
TCPIPJOBNAME 63, 148
verifying 148
TCPSTACKSOURCEVIPA 113, 128, 212, 213, 226
Telnet
accounting 40
associated printer function 347
connection and session takeover 356
connection security 319
connection types 314
device types 361
diagnostics 367
disconnect on error 360
Express Logon Feature (ELF) 361
generic connection requests 344
getting started 306
keep LU for client identifier 349
keeping the ACB open 360
logmode considerations 361
LU assignments 325
LU group capacity warning 349
LU mapping by application name 349
LU mapping selection rules 352
LU name assignment user exit 345
LUMAP statements, multiple 348
managing the server 309
map default application and ParmsGroup by LU group 347
mapping groups to client identifiers 345
mapping objects to client identifiers 325
mapping statements 335
overview 305
queuing sessions 358
session initiation management 354
SMF records 370
solicitor 362
specific connection requests 344
starting 306
storage considerations 307, 319
timers 366
TN3270 Enhanced (TN3270E) 315
TN3270 Telnet server, overview 305
Unformatted System Services (USS) 362
using wildcards to configure 306, 327, 737, 738
VTAM configuration data sets 306
Workload Manager (WLM) 372

Telnet *(continued)*
z/OS UNIX (otelnetd) 374
Telnet server, configuring 374
TESTSITE 152
TFTP server, configuring 408
TIMED daemon 699
TLS
for DCAS 721
for FTP server 721
for Telnet server 721
TLS, security 93
TN3270 Enhanced (TN3270E) 315
TN3270 Telnet server 305
TNF 72
trademark information 816
Traffic, splitting with VIPA 793
translating TCP/IP messages 76
translation of data, FTP 393
TRMD
running as a started task 620
running from the z/OS UNIX shell 620
stopping and starting 620
TRMDSTAT 621
TSIG 475

U

UCOUNT 390
UMASK 391
unique application-instance scenario 218, 219
UNITNAME 391
UNIX, superuser authorization 48
user interface
ISPF 811
TSO/E 811

V

variables, setting environment for resolver configuration files 29
VARSUBNETTING (IPCONFIG VARSUBNETTING) 113, 778
VCOUNT 391
verification
system configuration 76
X Window System 153
VIPA (virtual IP address)
backing up TCP/IP stack 215
configuring static 213
distributed VIPA 209
dynamic (DVIPA) 215
dynamic routing 209
dynamic VIPA 209
interfaces 185
manual movement 211
overview 65, 209
splitting traffic with 793
static 213
takeover planning 210, 216
VIPA interfaces 185
virtual IP address, see also VIPA 209
virtual machine communication facility, see also VMCF 72
VMCF (virtual machine communication facility)
commands 73
configuring as non-restartable system 73
configuring as restartable system 72
VOLUME 391
VPN, security 89
VTAM APPL definition
for SNALINK LU0 273
for SNALINK LU6.2 277
for X.25 NPSI 283
VTAM parameters, general update 75
VTAM, online information xxi
VTAMLST member 75

W

well-known procedure names, defining 654
wizard, for configuration 6
WLM (Workload Manager) 372
Workload Manager for Telnet 372

X

X Window System verification 153
X Windows, verifying installation 153
X.25 NPSI 278
cataloged procedure 280
configuration 280
configuration data set 280
configuring 279
DATE 280
DEVICE 279
GATE 280
GATEWAY 279
HOME 279
LINK 279
performance 279
START 279
VTAM definitions 283

Z

z/OS CS environment, overview 18
z/OS msys for Setup 6
z/OS UNIX initialization failure 71
z/OS UNIX System Services (z/OS UNIX)
applications and syslogd 107
concepts 10
Hierarchical File System (HFS) concepts 11
overview 10
z/OS UNIX Telnet server, configuring 374
z/OS UNIX, superuser authorization 48
z/OS, documentation library listing xxiii
z/OS, listing of documentation available 807
zone transfers 421

Communicating Your Comments to IBM

If you especially like or dislike anything about this document, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this document.
- If you prefer to send comments by FAX, use this number: 1-800-254-0206
- If you prefer to send comments electronically, use this network ID: usib2hpd@vnet.ibm.com

Make sure to include the following in your note:

- Title and publication number of this document
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

z/OS Communications Server
IP Configuration Guide
Version 1 Release 4

Publication No. SC31-8775-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

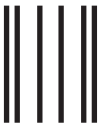
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC31-8775-02

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Software Reengineering
Department G7IA/ Bldg 503
Research Triangle Park, NC
27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5694–A01 and 5655–G52

Printed in U.S.A.

SC31-8775-02

Spine information:

z/OS Communications Server

z/OS V1R4.0 CS: IP Configuration Guide

Version 1
Release 4